Keystroke Level Model: Its concepts and application

Sarah A. Douglas

(From the book by Douglas & Mithal, *The Ergonomics of Computer Pointing Devices,* Springer-Verlag, 1997)

The GOMS model is very difficult to apply in practical situations where a usability analyst may wish to predict error-free performance time for a task. The Keystroke Level Model is Card, Moran and Newell's modification of GOMS to make it a more practical tool for designers. The Keystroke-Level Model (KLM) is a simplification of a GOMS representation at the keystroke level of basic perceptual, motor and cognitive operators (Card et al., 1980; 1983). Unlike GOMS, the representational components are given and a well-specified technique is described for creating a model of behavior for a specific task. As a derivation of the GOMS model, the KLM was intended to describe and predict only well-practiced, skilled behavior. However, unlike GOMS, the KLM predicts mean performance time for an average user to execute a specified task method; it does not attempt to predict sequences of methods and operators or to predict the cognitive time that it may take to acquire the task.

1.1.1 Description

A KLM analysis begins with the analyst selecting a task for which predicted task time is desired. Using the GOMS definition of unit-task, the time for the task can be written as the sum of the time to acquire the task and the time to execute it:

$$T_{unit\text{-}task} = T_{acquire} + T_{execute} \quad (1)$$

Acquisition times are not predicted by the KLM and depend on the types of task. For example, for manuscript editing from a marked-up page, it is about 2-3 seconds. For a CAD-CAM design task, it is about 5-30 seconds.
The execution time is then computed as a summation of the times of a fixed set of elementary operators. The basic operators are key stroking (K), pointing (P), homing (H), drawing (D), a mental operator (M), and system response time (R).

$$T_{execute} = T_K + T_P + T_H + T_D + T_M + T_R \quad (2)$$

Decisions must be made about translating observed behavior into these operators. Key stroking is the basic keying operation whether typing text characters or pressing a mouse button or function key. Pointing is using a pointing device—a mouse or joystick—to point to a target. Pointing time does not include pressing the mouse button for selection of a target. Homing is device switching time or moving the hands to or from the pointing device and keyboard. Drawing is using a mouse to define a drawing composed of a number of straight-line segments. The mental operator is the time for mentally preparing

to execute the previous physical operators. It involves decision-making time as well as recall or retrieval of the operators or method. System response time is required when the user must wait for the system to respond to a particular physical action. Further definition of these operators and their predicted performance time is given in Table 1. As the table demonstrates, the KLM recognizes the importance of pointing devices in GUIs. Of particular note is the fact that the Pointing operator is estimated at a mean of 1.1 seconds compared to 0.2 seconds for Keying—a factor of five times longer. In a graphical user interface, a great deal of time is spent pointing.

Table 1 Operators of the Keystroke-Level Model (from Card, Moran & Newell, 1983, p. 264)

Operator	**Description and Remarks**	**Time (sec)**
K	PRESS KEY OR BUTTON. Pressing the SHIFT or CONTROL key counts as a separate K operation. Time varies with the typing skill of the user. When unknown, use rate given of average skilled typist (55 wpm).	0.20
P	POINT WITH MOUSE TO TARGET ON DISPLAY. The time to point varies with distance and target size according to Fitts' law, ranging from .8 to 1.5 sec with 1.1 being an average. This operator does not include the (.2 sec) button press that often follows. Mouse pointing time is also a good estimate for other efficient analogue pointing devices, such as joysticks.	1.10
H	HOME HAND(S) ON KEYBOARD OR OTHER DEVICE.	0.40
D(n_D,l_D)	DRAW n_D STRAIGHT-LINE SEGMENTS OF TOTAL LENGTH l_D CM. This is a very restricted operator; it assumes that drawing is done with the mouse on a system that constrains all lines to fall on a square .56 cm grid. Users vary in their drawing skill; the time given is an average value.	0.9 n_D + 0.16 l_D
M	MENTALLY PREPARE.	1.35
R(t)	RESPONSE BY SYSTEM. Different commands require different response times. The response time is counted only if it causes the user to wait.	t

To illustrate the use of the KLM, an example task called, **Make_Labeled_Circle**, will be encoded. Figure 1 illustrates the desired drawing. The circle is drawn with a pattern and then labeled "Target". The task will be performed using an early version of the Macintosh MacDraw object-based graphics editor whose interface is shown in the figure.

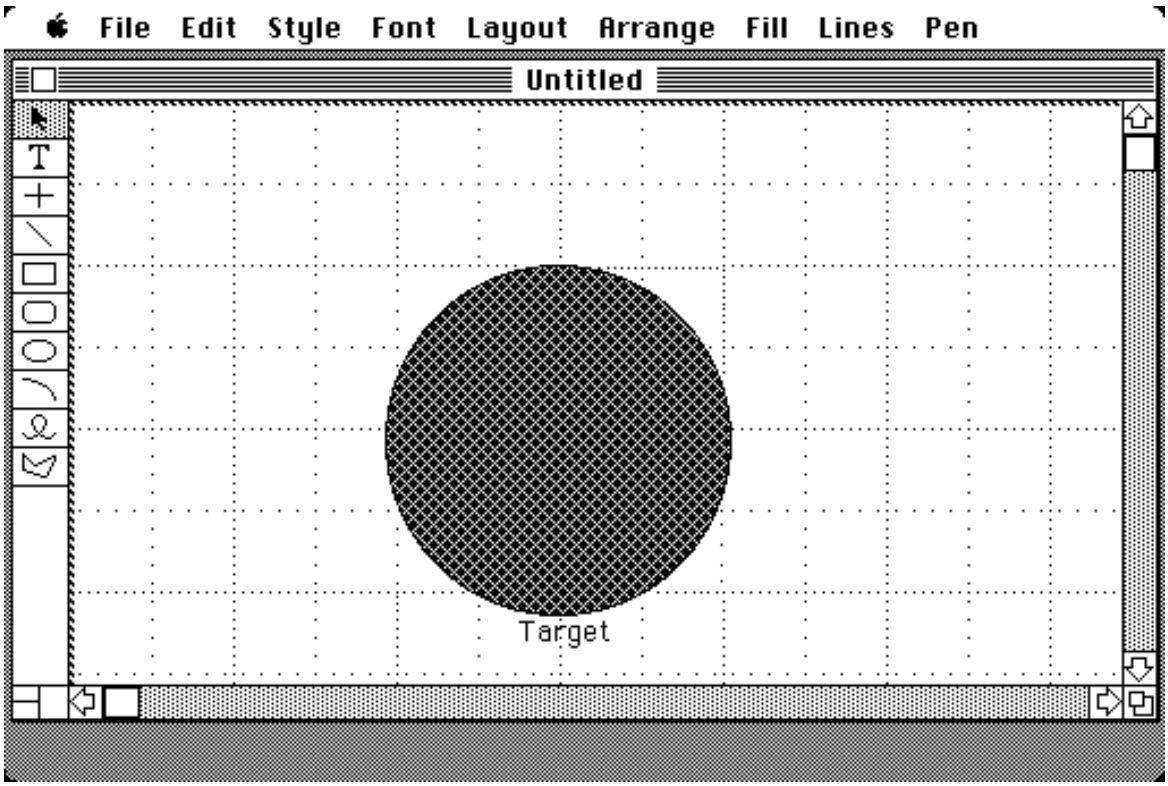

Figure 1 Make_Labeled_Circle Screen

The method selected for the **Make_Labeled_Circle** task is roughly described as the following unit-tasks:

Draw_Filled_Circle

Label_Circle

Unlike a GOMS representation, there is no notion of multiple methods for achieving a goal. The KLM lacks the GOMS control structure of Goal hierarchy or Selection rules for choosing between methods. Unlike GOMS, the KLM will only predict performance for a specific method, not the entire sequence of behavior given multiple tasks and methods. Taking the first unit-task, **Draw_Filled_Circle**, as an example, we write down the actions that the user will perform. Note that the SHIFT key must be held down when drawing a circle to distinguish it from an ellipse.

Draw_Filled_Circle

Reach for mouse from keyboard

Point to command to draw circle (icon looks like an ellipse)

Select icon (click mouse button)

Point to Fill on menubar

Select menu (press and hold mouse button down)

Drag (Pull Down) menu to Pattern menu item

Select Pattern menu item (release mouse button)

Press and hold down SHIFT key (for a true circle, not ellipse)

Point to position in drawing for perimeter of circle

Press and hold mouse button down

Draw diameter of circle (5 cm)

Release mouse button and SHIFT key

Next the task is written out using the elementary operators **K**, **P**, **H**, and **D**. Mental and system response operators are placed later.

Draw_Filled_Circle

Reach for mouse from keyboard	**H**
Point to command to draw circle	**P**
Select icon (click mouse button)	**K**
Point to Fill on menubar	**P**
Select menu (press and hold mouse button down)	**K**
Drag (Pull Down) menu to Pattern menu item	**P**
Select Pattern menu item (release mouse button)	~~**K**~~
Press and hold down SHIFT key	**K**
Point to position in drawing for perimeter of circle	**P**
Press and hold mouse button down	**K**
Draw diameter of circle (5 cm)	**D**
Release mouse button and SHIFT key	~~**K**~~

This is the sequence of physical actions to achieve **Draw_Filled_Circle**. A minor decision had to be made about how to handle the pull-down menu. The analyst decided that the dragging should be represented by a pointing and two keying operators for the mouse button press, hold, and release. The original KLM does not specify how to handle pull-down menu operations or dragging. We have also introduced a ~~**K**~~, which signifies

that there is a keystroke, i.e. a released key, but that its time is counted when the key is originally pressed down. Note also that there is no way in the KLM to handle a differently handed, parallel release of both the mouse button and the SHIFT key on the keyboard, except as a single action. In other words, both the KLM and GOMS can only describe one stream of sequential actions.

There are no waiting periods for system response, but the mental operators need to be added. Table 2 is the list of heuristic rules used for placing mental operators. Card et al. developed these rules as heuristics since they could not discover theoretically or empirically a deterministic (causal) method for locating where a cognitive "action" occurs in a pattern of physical and perceptual actions.

Table 2 Heuristic rules for placing the M operators (From Card et al., 1983, p. 265)

Begin with a method encoding that includes all physical operations and response operations. 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 that are not part of 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., **PMK⇒ PK**).
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.
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.

Beginning with heuristic Rule 0 given in Table 2, the analyst places candidate mental operators in the method where K's and P's are located. Rule 0 places mental operators in front of all K's that do not enter text or numbers. In our example, an M is placed before a mouse button K. (The ~~K~~'s are ignored.) Using Rule 0 requires that a decision be made about the structure of the commands since M's are placed in front of all P's that select commands (actions). We have decided that there are three separate commands in this method: selecting an icon command, selecting a pattern, and drawing a circle.

Draw_Filled_Circle

Reach for mouse from keyboard	**H**
Select command icon:	
Point to command to draw circle	**MP**
Select icon (click mouse button)	**MK**
Select pattern:	
Point to Fill on menubar	**MP**
Select menu (press and hold mouse button down)	**MK**
Drag (Pull Down) menu to Pattern menu item	**P**
Select Pattern menu item (release mouse button)	~~**K**~~
Draw circle:	
Press and hold down SHIFT key	**MK**
Point to position in drawing for perimeter of circle	**P**
Press and hold mouse button down	**MK**
Draw diameter of circle (5 cm)	**D**
Release mouse button and SHIFT key	~~**K**~~

An alternative parsing is possible. The entire method could be seen as one action (command): the draw circle command. This single command has two operands: one to select the pattern, and another to specify the parameters of the size of the circle. If that were the case, then there would only be one mental operator at the beginning of the pointing operator for selecting the draw circle icon. The KLM was originally developed for simple typed command languages, which usually had a simple syntactic structure of action [object, modifiers]. GUI interface syntax may often specify the object first in the syntax and then the command. Command specification may also be much more complex than typing in two or three characters, such as selecting a command from a pull-down menu.

After placing all candidate M's in the method, Rules 1 to 4 are applied. In the **Draw_Filled_Circle** method Rule 1 is applied to delete three M's from the mouse button press after pointing, since the key presses are fully anticipated in the Pointing operator. The method now has the following operators:

Draw_Filled_Circle

Reach for mouse from keyboard	**H**
Select command icon:	
Point to command to draw circle	**MP**
Select icon (click mouse button)	~~**M**~~**K**

Select pattern:	
Point to Fill on menubar	**MP**
Select menu (press and hold mouse button down)	**~~M~~K**
Drag (Pull Down) menu to Pattern menu item	**P**
Select Pattern menu item (release mouse button)	**~~K~~**
Draw circle:	
Press and hold down SHIFT key	**MK**
Point to position in drawing for perimeter of circle	**P**
Press and hold mouse button down	**~~M~~K**
Draw diameter of circle (5 cm)	**D**
Release mouse button and SHIFT key	**~~K~~**

Substituting in equation 2, the predicted time to execute this method is:

$$T_{execute} = 4T_K + 4T_P + 1T_H + 1T_D + 3T_M \quad (3)$$

Multiplying by the times given in Table 1 is:

$$T_{execute} = (4 * 0.2) + (4 * 1.1) + (1 * 0.4) + (0.9 + (0.16 * 5\ cm)) + (3 * 1.35) \quad (4)$$

For a predicted total time of:

$$T_{execute} = 11.35\ secs \quad (5)$$

The KLM technique is applied to the second unit-task, **Label_Circle**:

Label_Circle

Select command icon:	
Point to command to create text (icon looks like a T)	**MP**
Select icon (click mouse button)	**~~M~~K**
Position label:	
Point to position in drawing for beginning of text	**MP**
Select position (click mouse button)	**~~M~~K**
Reach for keyboard from mouse	**H**
Type word “Target”	**6K**

Substituting in equation 2, the predicted time to execute this method is:

$$T_{execute} = 8T_K + 2T_P + 1T_H + 2T_M \quad (6)$$

Multiplying by the times given in Table 1 is:

$$T_{execute} = (8 * 0.2) + (2 * 1.1) + (1 * 0.4) + (2 * 1.35) \quad (7)$$

For a predicted total time for labeling the circle of:

$$T_{execute} = 6.9 \text{ secs} \quad (8)$$

The task execution time for the whole task of drawing and labeling the circle is predicted at:

TOTAL EXECUTION time = 18.25 seconds

by summing the two unit-task execution times (Equations 5 and 8). There is some variability in writing a KLM method as discussed above, so there are really a range of predictions depending on where M's are placed, etc. Variability from variance in users' behavior is also ignored. (The times for operators in the KLM can be considered means of a normal distribution.) We have also ignored the task acquisition time, which is not predicted by the KLM, but must be estimated by the analyst. In the next section, we discuss how the KLM was validated and how our prediction for the **Make_Labeled_Circle** task compares to experimentally collected data.

1.1.2 Validating the KLM

A validation study was done by Card et al. (1980, 1983) using 1280 user-system-task interactions. The data came from 28 different participants, 10 different computer systems, and 14 tasks. The systems included three different text editors (four tasks each), three different graphics editors (five tasks each), and five different file management systems (one task each). For each task-system combination, the most "natural" method was coded in KLM operations and total task time computed. This predicted time was compared to the observed time of the participants. The average RMS error for all tasks was 21%, about the same as the prediction error for the GOMS model at the keystroke level. Since the standard (sampling) error[1] was about 9%, the rest of the prediction error,

[1] Standard error (SE) of estimation of the population mean for samples of size N. SE=SD/√N.

i.e. 12%, is due to a failure of the model to accurately predict. The average coefficient of variation[2] for the tasks is .31 which is expected for tasks averaging about 8 seconds in duration and composed of sequences of operators (Abruzzi, 1952; 1956). The KLM appears to be a fairly robust performance model for predicting well-practiced, skilled, error-free performance time for tasks whose operators, including pointing operators, fit the empirical constraints of the model.

REFERENCES:

Abruzzi, A. (1956). *Work, Workers and Work Measurement*. New York: Columbia University Press.

Card, S. K., Moran, T. P. and Newell, A. (1980). The Keystroke Level Model for User Performance Time with Interactive Systems. *Communications of the ACM, 23*, pp. 396-410.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Douglas, S.A. and Mithal, A.K. *The Ergonomics of Computer Pointing Devices*. Springer-Verlag, 1997.

[22] Coefficient of Variation or CV = SD / mean. Coefficient of variation partially normalizes the standard deviation to make it more comparable for operators of different durations.