# Physical and Device models

## The Keystroke-Level Model

The Keystroke-Level Model (KLM) was developed by Card, Moran & Newell in 1980.

It is designed to model unit-tasks within an interaction.

These would typically be short command sequences, such as changing the font of a character.

The KLM would rarely be used to model sequences lasting more than twenty seconds.

The Keystroke-Level Model divides tasks into two phases:

* **Acquisition** - the user builds a mental model of the task.
* **Execution** - the task is executed using the system's facilities.

The KLM does not attempt to model what happens during the acquisition phase.

This must be done using other models or methods.

However, the KLM models what happens during the execution phase in great detail.

The execution phase is broken down into **physical motor operations**, **system responses**, and **mental operations**.

The KLM defines five types of motor operation:

|  |  |
| --- | --- |
| **K** | keystroking, i.e., striking a key, including a modifier key such as shift |
|   |   |
| **B** | Pressing a mouse button |
|   |
| **P** | Pointing, using the mouse or other pointing device, at a target |
|   |
| **H** | Homing, i.e., switching the hand between mouse and keyboard |
|   |
| **D** | Drawing lines using the mouse |

The KLM also provides mental response and system response operators:

|  |  |
| --- | --- |
| **M** | Mentally preparing for a physical action |
|   |   |
| **R** | Response from the system: may be ignored in some cases, e.g., copy-typing |

Suppose we wish to model the interaction involved in correcting a single-character error using a mouse-driven text-editor.

This involves pointing at the error, deleting the character, re-typing it, then returning to the original point in the text.

This might be modelled as follows:

|  |  |  |
| --- | --- | --- |
| 1 | move hand to mouse | **H**[mouse] |
|   |
| 2 | position cursor after bad character | **PB**[LEFT] |
|   |
| 3 | return hand to keyboard | **H**[keyboard] |
|   |
| 4 | delete character | **MK**[DELETE] |
|   |
| 5 | type correction | **K**[char] |
|   |
| 6 | reposition insertion point | **H**[mouse]**MPB**[LEFT] |

Once an operation has been decomposed in this way, the time required to perform it can be calculated.

This is done by counting the number of each type of operation, multiplying by the time required for each type of operation, then summing the times, e.g.:

*Texecute* = *T*K + *T*B + *T*P + *T*H + *T*D + *T*M + *T*R

For example, the time required for the operation described earlier could be calculated as follows:

*Texecute* = *2t*B + *3t*H + *2t*K + *2t*M + *2t*P

The times for each operation would be obtained from empirical data:

* Keying (**K**) time depends upon the skill of the user, and would vary for different types of user.
* The time required to press a mouse-button (**B**) varies little between users
(hence it is useful to be able to specify different times for button-presses and keystrokes)
* Pointing time (**P**) can be calculated using Fitt's Law
* Homing time (**H**) is assumed to be a constant.
* Drawing time (**D**) is highly domain-specific and would have to be determined for a particular system and application.

The system response time (**R**) can be determined by measurements on the system.

The mental preparation time operator (**M**) is probably the most complex to use.

KLM assumes that the user has decided what to do *before* embarking on an operation (the acquisition phase).

Thus the mental operator represents only the time taken to recall the decision, not the time taken to make the decision.

This makes it possible to define a unit time for mental operations - the period is that needed for recall from short-term memory.

Instead of having to determine *how long* a particular mental task might take, a KLM-user only has to work out *how many* recall operations are involved.

The KLM incorporates a comprehensive set of heuristics for determining where recall operations are required.

These are largely based on the concept of *chunking*.

For example:

* a well-known and/or frequently-used command-name might be stored as one chunk.

Hence it could be recalled in one **M** operation.

* an unfamiliar word or command-name might be stored with each character occupying a separate chunk of memory.

Hence it would require a separate **M** operation for each character.

However, note that chunking strategies vary depending upon the skill, general knowledge, and domain knowledge of the user.

Thus the heuristics only provide an approximation.

Note that it's not important to use exact times for each operation when calculating the total execution time.

It's more important that the *relative* timings be correct.

This allows the effects of different interaction methods and styles to be compared reliably.

The KLM has been used to model short interaction sequences on many systems, and has yielded useful comparisons.

Many of the interaction methods and devices in use today were tested and refined using this model.

Card, Moran & Newell tested the KLM using a range of systems, both CLI- and GUI-based. They also used it to make predictions about hypothetical systems.

They claim that it analysed execution times with an error of around 20% (when compared with empirical findings).

This claim has been confirmed by other researchers.

The KLM is therefore one of the few models capable of providing accurate quantitative predictions regarding performance.

However, note that:

* The KLM can only be used for short sequences of commands - micro-interactions.

It cannot be guaranteed that the results will be as reliable when the KLM is used in conjunction with higher-level models to analyse larger interactions or complete systems.

* The KLM only provides a measure of execution time - but fastest is not always best.

The KLM does not claim to measure factors such as cognitive load, frustration, etc..

## KLM Example

As a simple example of the use of the KLM, consider the operation of closing a window using either:

* a function key
* a menu option.

At the start of the operation, the user's hand is assumed to be on the mouse.

Closing the window using the function key can be modelled as follows:

**H**[to keyboard] **MK**[function key]

Closing the window by selecting 'close' from the menu can be modelled as follows:

**P**[to menubar] **B**[LEFT down] **MP**[to option] **B**[LEFT up]

To calculate the relative timings we need timing values for the individual operations.

Card, Moran & Newell derived the following timings from their experiments:

|  |  |  |  |
| --- | --- | --- | --- |
| **K** | Key-press | 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 |
|   |   | complete click | 0.20 |
|   |  |  |
| **P** | Pointing with mouse | Fitt's law | use formula |
|   |   | average | 1.10 |
|   |  |  |
| **H** | Homing hands to and from keyboard | 0.40 |
|   |  |  |
| **D** | Drawing | domain-dependent |
|   |  |  |
| **M** | Mental preparation | 1.35 |
|   |  |  |
| **R** | System response | system-dependent |

Applying these values to the models gives the following timings.

For the function-key operation:

|  |  |  |
| --- | --- | --- |
|   | *Texecute* | = *t*H + *t*M + *t*K |
|   |   | = 0.4 + 1.35 + 0.28 |
|   |   | = 2.03 seconds |

For the menu-selection operation:

|  |  |  |
| --- | --- | --- |
|   | *Texecute* | = *2 t*P + *2 t*B + *t*M |
|   |   | = 2.2 + 0.2 + 1.35 |
|   |   | = 3.75 seconds |

It can be seen that, when modelled using the KLM, the function-key approach is significantly faster than using a mouse to select the option from a menu.

However, note that this figure is obtained using an *average* value for the pointing operation involved in menu-selection.

A more accurate figure can be obtained using Fitt's Law.

For the first pointing operation (moving the mouse onto the menu) assume that:

* The window is 300 pixels high, plus the menu
* The menu is at the top of the window and is 15 pixels high
* The cursor is in the middle of the window, directly below the menu

Therefore the distance to travel is 150 pixels and the distance-to-target-size ratio is 10:1.

For the second pointing operation (selecting the 'close' option from the menu) assume that:

* The 'close' option is the fourth item down
* All menu items have the same height

Therefore the distance-to-target-size ratio is 4:1.

However, note that this is *dragging* operation, not a *pointing* operation.

Inserting these figures into Fitt's Law, and using the appropriate values for *a* and *b* derived by Mackenzie, Sellen and Buxton (1991), we obtain the following timings:

|  |  |  |
| --- | --- | --- |
| **P**[to menubar] | = -107 + 223 *log2* (10+1) | = 664 milliseconds |

And:

|  |  |  |
| --- | --- | --- |
| **P**[to option] | = 135 + 249 *log2* (4+1) | = 713 milliseconds  |

Inserting these figure into the equations shown earlier gives a revised timing figure of 2.9 seconds.

This is much closer to the figure obtained for using the function key, which is 2.03 seconds.

Using the same figures, we can also calculate best-case and worst-case times for menu-selection.

We will assume the same window size, menu-arrangement, etc..

* The time required for the first operation (pointing) will vary depending upon the initial mouse-position:
	+ If the mouse is right next to the menu at the start of the operation, the distance will be zero and hence the selection-time will be zero.
	+ If the mouse is at the bottom of the window at the start of the operation, the distance will be 300 and the selection time will be:

-107 + 223 *log*2 (300/15+1) = 872 milliseconds

* The time required for the second operation (dragging) does not change: it is still 713 milliseconds

Inserting these figure into the equations shown earlier gives timings of:

* Best case - 2.26 seconds
* Worst case - 3.14 seconds