CHAPTER

Modeling Interaction 7

A model is a simplification of reality. Consider an architect's scale model of a building or a physicist's equation for the trajectory of a tossed ball. Both are reductions or simplifications of more complex phenomena. They are useful because they allow us to explore the phenomena, think about them, make changes, and so on, without actually constructing the building or throwing the ball. A great many problems in HCI are explored in this manner. This chapter is about modeling interaction—building models, testing models, using models, and thinking about interaction through models.

The term *model* is often used loosely, without a clear and simple definition. A mathematician's model is probably quite distant from a psychologist's or sociologist's model. To the mathematician, a model is a formal calculus tested through computer simulation. To the psychologist or sociologist it is often a verbal-analytic description of behavior. Pew and Baron (1983) elaborate on this: "There is a continuum along which models vary that has loose verbal analogy and metaphor at one end and closed-form mathematical equations at the other" (p. 664). This seems like a useful way to organize our discussion. Models using "loose verbal analogy and metaphor" describe phenomena. Let's called them *descriptive models*. Models using "closed-form mathematical equations" predict phenomena. Let's call them *predictive models*. This chapter opened with an example of each. An architect's model of a building is a descriptive model. A physicist's equation for the trajectory of a tossed ball is a predictive model.

There are many examples of descriptive and predictive models in HCI. The rest of this chapter is organized in two parts. In the next section I present descriptive models. Examples are presented, along with discussion on how each can provide insight into a design or interaction problem. Following this, I present a few examples of predictive models, with a similar organization.

7.1 Descriptive models

Descriptive models are everywhere. They emerge from a process so natural it barely seems like modeling. Look in any HCI paper with a title or section heading using words like *design space*, *framework*, *taxonomy*, or *classification* and there is a good chance you'll find a descriptive model, perhaps without knowing it. In many cases, the word *model* isn't even used. The general idea is developed in the next section.

7.1.1 Delineating a problem space

A descriptive model can be as simple as a dividing up a problem space. Taken as a whole, without divisions, the problem space is … well, that's what it is, a space—vast and uncharted, a big fuzzy cloud. However, with a little thought and organization it becomes a partitioned domain. As a partitioned domain, we are empowered to think differently about the problem space, to get inside it and see the constituent parts or processes. We can focus on certain parts of the problem space, consider how one part differs from, or relates to, another, and weigh strengths, weaknesses, advantages, or disadvantages of certain parts over others.

Here is a non-HCI example, just to get started. Consider *politics*, a subject we all know a little about. Figure 7.1a gives the "big fuzzy cloud" model of politics. Of course, there is no model; it's just the thing itself, without delineation. If we really want to study politics, it would be useful to break it down, delineate it, categorize it, structure it, or whatever—so we can get inside the problem, define and understand its constituent parts, and begin the process of charting out a corner of the problem space as a research area. In Figure 7.1b, we see Johnson's (2007, 19)

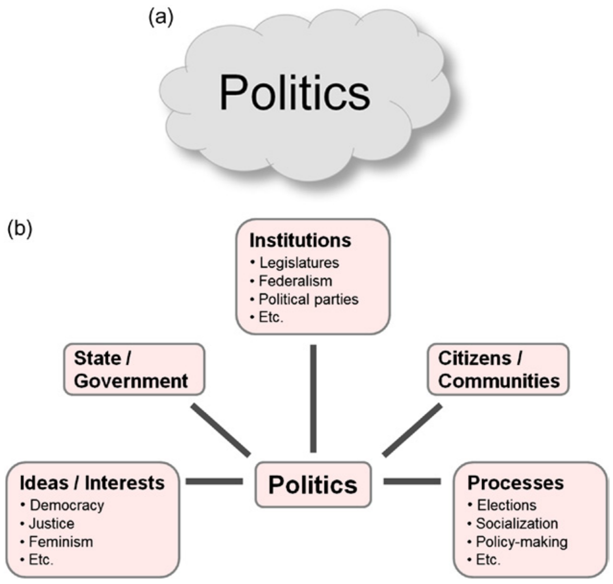


FIGURE 7.1

Politics: (a) The "big fuzzy cloud" model. (b) A descriptive model delineating the problem space.

delineation of the problem space for politics. This is a descriptive model for politics. Open any textbook on any subject and you are likely to find similar diagrams.

Johnson's chart in Figure 7.1b is beautifully crafted. It is organized as a semi-circular spoked wheel with politics at the center. Do you like the organization? Is there a different organization that might work better? Note the symmetry between "State/Government" (left) and "Citizens/Communities" (right). Nice. "Ideas/Interests" is mirrored with "Processes." Is that reasonable? What about "Federalism" under Institutions? That seems odd. Is federalism an institution? Perhaps Federalism, like Democracy, should be under Ideas/Interests. Enough said. The descriptive model in Figure 7.1b adds tremendous insight into politics. With it, we are empowered not only to think differently about the problem, but to think critically about it. With every tweak, refinement, and improvement we get a better model and, more importantly, we get a better understanding of the problem. That's the power of descriptive models.

In HCI, researchers often approach problems in a similar way. Let's have a look at a few descriptive models in HCI and examine how they are used to analyze a problem space and inform interaction design.

7.1.2 Quadrant model of groupware

An important area of research within HCI is computer supported collaborative work (CSCW). Computer applications to support collaboration are known as *groupware*. There are many facets to groupware, including teleconferencing, team writing, group voting, and so on. Is it possible to delineate the problem space of groupware into a descriptive model? Of course it is. And there are many ways to do this. The starting point, as always, is to just think about groupware, to consider aspects of it that differ in some way. These differences can assist in dividing the problem space into simpler components or processes.

One such possibility for groupware is to consider collaboration in terms of space and time. Spatially, users might collaborate in the same physical place or in different physical places. Temporally, users might collaborate at the same time or at different times. The spatial and temporal aspects of the collaboration are illustrated in a descriptive model known as the *quadrant model of groupware* (Johansen, 1991; Kaplan, 1997). (See Figure 7.2.) The model presents groupware as a 2×2 matrix with four cells, each representing a distinct quality of interaction. A group of colleagues co-authoring a report involves people working in different places and different times. This is labeled *group writing* and appears in the bottom-right cell. A team meeting where members use computer technology to present and discuss ideas involves people working in the same place and at the same time. This is labeled *PC projectors* and appears in the top-left cell.

The examples in Figure 7.2 are from a report published in 1991 (Johansen, 1991). There are numerous forms of collaboration common today that didn't exist in 1991. Think of camera phones, web cams, social networking, tweeting, and so on. All involve people collaborating or interacting with other people in some

	Same Time	Different Times
Same Place	Copy boards PC Projectors Facilitation Services Group Decision Room Polling Systems	Shared Files Shift Work Kiosks Team Rooms Group Displays
Different Places	Conference Calls Graphics and Audio Screen Sharing Video Teleconferencing Spontaneous Meetings	Group Writing Computer Conferencing Conversational Structuring Forms Management Group Voice Mail

FIGURE 7.2

Quadrant model of groupware.

(Adapted from Johansen, 1991, Figure 1)

manner. Can these interactions be positioned in the quadrant model of groupware? Probably. One use of a descriptive model is to identify common elements for interactions located in the same area of the problem space. Perhaps all such interactions should contain the same common elements, and if they don't, this might motivate a design change. The goal here is not to engage in this exercise for groupware, but only to suggest that descriptive models provide a context for this sort of analysis. The quadrant model of groupware seems to offer this possibility.

7.1.3 Key-action model (KAM)

Here is another descriptive model that might be useful in HCI. Computer keyboards today contain a vast array of buttons, or keys. Most desktop systems use a keyboard with a row of function keys across the top and a numeric keypad on the right. Have you ever thought about the operation and organization of keys on a keyboard? Here's a descriptive model for this purpose. We'll call it the *key-action model* (KAM). With KAM, keyboard keys are categorized as either *symbol keys*, *executive keys*, or *modifier keys*. Symbol keys deliver graphic symbols to an application such as a text editor. These are typically letters, numbers, or punctuation symbols. Executive keys invoke actions in the application or at the system level. Examples include ENTER, F1, and ESC. Modifier keys do not generate symbols or invoke actions. Instead, they set up a condition that modifies the effect of a subsequently pressed key. Examples include SHIFT and ALT. That's about it for the KAM. It's a simple model. It has a name, it delineates a problem space, and it identifies three categories of keys; for each category it provides a name, a definition, and examples. I have not proposed a chart like in Figure 7.1b for politics. Perhaps you can do that. What do you think of KAM? Is it correct? Is it flawed? Do all keyboard keys fit the model? Can you think of additional categories or sub-categories to improve the model or to make it more accurate or more comprehensive? Do some keys have features of more than one category? Is the model useful?

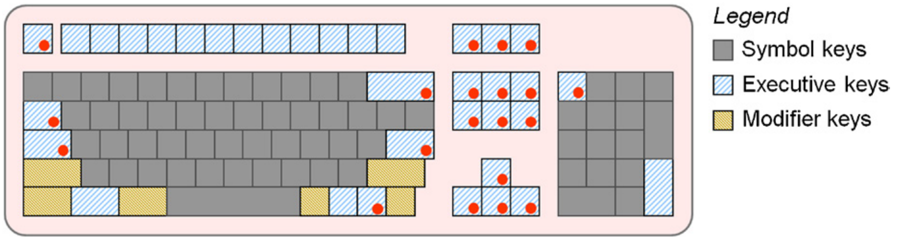


FIGURE 7.3

The key-action model (KAM) illustrated. Keys marked with (red) dots are executive keys that are not mirrored on both sides.

The questions above—by their very nature—are evidence of the power of descriptive models such as the KAM. The model piques our interest and suggests aspects of keyboard operation that merit consideration, particularly if a new design is contemplated. The most important question is the last. There is no greater measure of the merit of a model than its ability to tease out critical arguments on the potential, the capabilities, and the limitations in an interaction domain. Can the KAM do that? Let's see.

Figure 7.3 illustrates a keyboard with keys highlighted according to the KAM. It's a typical desktop keyboard with a wide space bar along the bottom, function keys along the top, a numeric keypad on the right, and non-alpha keys in various locations.

Let's think about the organization of keys in Figure 7.3 in terms of left- and right-hand usage. First, consider the executive keys (e.g., ENTER) and modifier keys (e.g., SHIFT).[1] On the keyboard's left we find seven such keys: SHIFT, ALT, CTRL, TAB, CAPS_LOCK, ESC, and WINDOWS. On the right we find no less than 22: SHIFT, ALT, CTRL, ENTER (X2), WINDOWS, RIGHT_CLICK, BACKSPACE, INSERT, DELETE, HOME, END, PAGE_UP, PAGE_DOWN, ←, ↑, →, ↓, PRNT_SCRN, SCROLL_LOCK, PAUSE, and NUM_LOCK. Because SHIFT, CTRL, ALT, and WINDOWS are mirrored, they do not pose a left- or right-hand bias and are eliminated from further discussion. Only three keys on the left (ESC, TAB, CAPS_LOCK) are without a right-side replica, thus, the numbers are three on the left, 18 on the right. These are identified by dots in Figure 7.3.

Before continuing, let's remember our goal. I am not here is to deliver a tutorial on keyboards, but to develop a descriptive model and demonstrate the ability of the model to delineate a problem space and potentially expose problems and suggest opportunities. The analysis in the preceding paragraph is a good example of this. Using the key-action model, we have identified a very peculiar bias in desktop keyboards. Let's continue.

With a 3:18 left-right ratio of executive keys, the desktop keyboard is clearly entrenched with a right-side bias. Simply put, the right hand is busy. Furthermore,

[1] For this discussion, we ignore the 12 function keys across the top.

with the emergence of the mouse in the 1980s, the right hand is even busier. Is it possible the right hand is, in fact, overloaded? Interactions that juxtapose executive-key activation and point-click operations are problematic for right-handed users. The right hand is just too busy. If the right hand is gripping the mouse and there is a need to press a right-side executive key, the options are to "reach over" with the left hand, or to release the mouse and acquire and activate the power key with the right hand. Not so good in either case. For users who manipulate the mouse with their left hand, the situation is quite different.[2] For "lefties," mouse operations efficiently mix with right-side power-key operations. In fact, a task analysis of common GUI tasks reveals an interesting phenomenon: the desktop interface is biased to favor left-hand mouse usage! But that's another story. Details are provided elsewhere (MacKenzie, 2003).

The key-action model only captures one aspect of keyboards, namely the actions associated with each key. Another way to think about keyboards is in the ambiguity of key presses. If pressing a symbol key always produces the same symbol, then the situation is simple. But if a key can produce two or more symbols, then this is worth thinking about. For this, a "key-ambiguity" descriptive model might be useful (MacKenzie and Soukoreff, 2002).

The key-action model delineates a design space for keyboards and allows for analyses to tease out design issues. Let's move on to another descriptive model.

7.1.4 Model of bimanual control

Humans are not only two-handed, they use each of their hands differently. This human behavior has undergone considerable study in a specialized area of human motor control known as *bimanual control* or *laterality* (Kelso, Southard, and Goodman, 1979; Peters, 1985; Porac and Coren, 1981; Wing, 1982). Studying the between-hand division of labor in everyday tasks reveals that most tasks are asymmetric: Our hands work together but have different roles and perform different tasks. Given this, and the knowledge that people are either right-handed or left-handed, examining the assignment of tasks to hands is a useful exercise. Guiard undertook such an exercise and proposed what he described as "a simple model, based on a non-quantitative physical approach, which aims at describing the logic of the division of labor that appears to govern the variety of human bimanual asymmetrical actions" (1987). This is the essence of a descriptive model. The result is Guiard's *model of bimanual control*—a simple three-part descriptive model identifying the roles and actions of the non-preferred and preferred hands. (See Figure 7.4.)

The points in Figure 7.4 are best explained through an exemplary illustration and narrative. In Figure 7.5, a right-handed graphic artist is sketching the design of

[2]Numerous straw votes by the author at presentations and lectures suggest that the vast majority of left-handed users manipulate the mouse with their right hand. The reason, it seems, is that most users learned to use a computer at an early age at school, where the mouse is positioned, and often anchored, on the right side of the keyboard.

Hand	Role and Action
Non-preferred	• Leads the preferred hand • Sets the spatial frame of reference for the preferred hand • Performs coarse movements
Preferred	• Follows the non-preferred hand • Works within established frame of reference set by the non-preferred hand • Performs fine movements

FIGURE 7.4

Guiard's model of bimanual control.

(From Guiard, 1987)

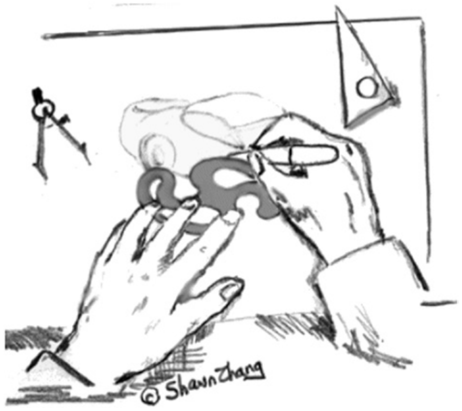


FIGURE 7.5

Two-handed interaction paradigm.

(Sketch courtesy of Shawn Zhang)

a new car. The artist acquires the template with the left hand (*non-preferred hand leads*). The template is manipulated over the workspace (*coarse movement, sets the frame of reference*). The stylus is acquired in the right hand (*preferred hand follows*) and brought into the vicinity of the template (*works within frame of reference set by the non-preferred hand*). Sketching takes place (*preferred hand makes precise movements*).

The roles and actions just described provide a provocative and fresh way of describing how humans approach common tasks. This is true both for every-day tasks and in the specialized context of human-computer interaction.

Guiard's research was published in a journal article in experimental psychology, not HCI. As originally published, no context for HCI was provided. Watershed

Task	Characteristics
Scrolling	• Precedes/overlaps other tasks • Sets the frame of reference • Minimal precision needed (coarse)
Selecting, editing, reading, drawing, etc.	• Follows/overlaps scrolling • Works within frame of reference set by scrolling • Demands precision (fine)

FIGURE 7.6

Relationship between scrolling and common GUI tasks.

moments in multi-disciplinary fields like HCI often occur when researchers, through due diligence, locate and adopt relevant research in other fields—research that can inform and guide their own discipline.[3] Guiard's work was located and studied by Paul Kabbash, a graduate student at the University of Toronto in the early 1990s. The paper by Kabbash, Buxton, and Sellen (1994) was the first in HCI to cite Guiard's 1987 work and adapt it to analyze and inform the problem space of two-handed computer input. Since then, Guiard's model has been widely adopted for HCI research in two-handed interaction (e.g., Cutler, Fröhlich, and Hanrahan, 1997; G. W. Fitzmaurice, Ishii, and Buxton, 1995; G. Kurtenbach, Fitzmaurice, Baudel, and Buxton, 1997; Malik and Laszlo, 2004; Morris, Huang, Paepcke, and Winograd, 2006; Yee, 2004). An example follows.

Scrolling is traditionally accomplished by dragging the *elevator* of the scrollbar positioned along the right-hand side of an application's window. Acquiring the elevator is a target acquisition task taking up to two seconds per trial. However, this action is in conflict with a basic goal of good user interfaces: unobtrusiveness and transparency. That is, users should not be required to divert their attention from the primary task (reading, editing, drawing, etc.) to acquire and manipulate user interface widgets.

Desktop affordances for scrolling changed dramatically in 1996 with the introduction of Microsoft's IntelliMouse, which included a scrolling wheel between the mouse buttons. Numerous copycat variations appeared afterward from other manufacturers. The so-called "wheel mouse" puts scrolling in the preferred hand. This is arguably bad for right-handed users because it increases the right-side bias noted in the preceding section. This insight—revealed in Guiard's descriptive model of bimanual control—presents an opportunity for design. An analysis of scrolling and the accompanying tasks reveals that scrolling is well suited to the non-preferred hand. Evidence of this is presented in Figure 7.6, which juxtaposes the properties of scrolling with tasks typically performed in concert with scrolling. (The reader is invited to compare the organization of bullets in Figure 7.6 with that in Figure 7.4, which presents the guiding principles in Guiard's model of bimanual control.)

[3]There are examples in HCI, for example, Card, English, and Burr's (Card et al., 1978) first use of Fitts' law (Fitts, 1954), or Norman's (1988) introduction of Gibson's affordances (1979).

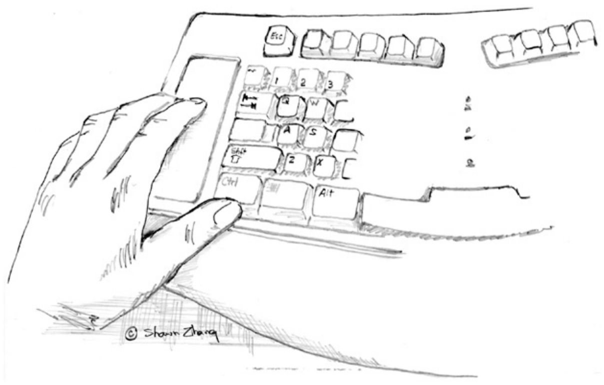


FIGURE 7.7

Scrolling interface example.

(Sketch courtesy of Shawn Zhang)

Figure 7.7 presents a scrolling concept for a right-handed user. On the keyboard's left a touch strip is shown, but a wheel is just as appropriate. There are many implementation issues, such as scrolling sensitivity and support for up/down paging, but these are not explored here. The point is simply that scrolling is an appropriate task for the non-dominant hand and that this is revealed through Guiard's model of bimanual control. See also Buxton and Myers (1986).

On a personal note, the idea of non-dominant hand scrolling was sufficiently appealing that I decided to re-engineer a Microsoft Intellimouse, separating the wheel assembly from the mouse. The new form factor allowed operation of the Intellimouse with the dominant hand while scrolling with the non-dominant hand. Figure 7.8a shows the setup for right-handed use. This setup was used for numerous demos and presentations. Two such presentations were given in February 1998 during a visit to Microsoft in Redmond, Washington. The first audience was the "mouse group" of Microsoft's Hardware Ergonomics Group. The idea seemed provocative enough. An impromptu meeting was arranged for the afternoon of the same day and the presentation was given again to the "keyboard group." The idea of non-dominant hand scrolling, as suggested by Guiard's model of bimanual control, was convincing. About two years later, Microsoft released the Office Keyboard, shown in Figure 7.8b. On the left side, the device includes some power keys and a scrolling wheel (see arrow)—perfect for right-handed users. Variations of keyboards with left-side scrolling were subsequently released by Microsoft and other keyboard manufacturers, including Logitech. In the end, however, the momentum and popularity of the wheel mouse were just too much. The benefit of non-dominant hand scrolling was insufficient to sway a satisfied user community toward a new interface paradigm. Today, most computer mice continue to include a wheel for scrolling.

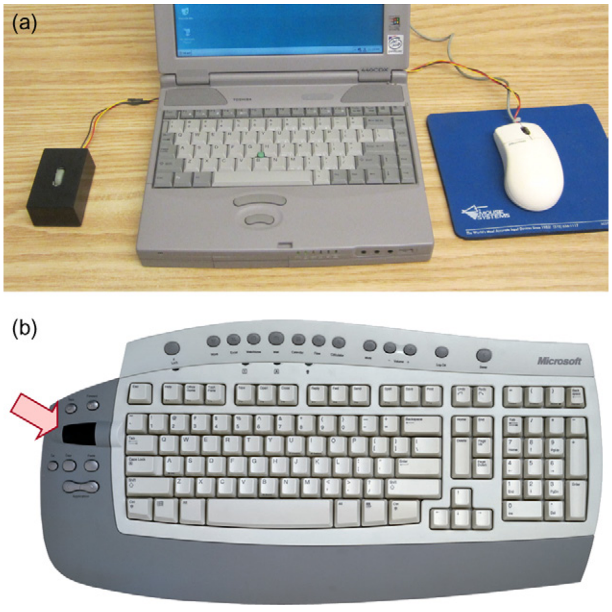


FIGURE 7.8

(a) The author's re-engineered Microsoft Intellimouse with the wheel on left side of the keyboard. (b) Microsoft Office Keyboard with scrolling wheel on the left side.

7.1.5 Three-state model for graphical input

Another descriptive model is Buxton's *three-state model of graphical input* (Buxton, 1990). The model is a simple characterization of the operation of computer pointing devices in terms of state transitions. It is described as "a vocabulary to recognize and explore the relationship between pointing devices and the interaction techniques they afford" (Buxton, 1990, p. 449). In this sense, it is a paradigm of descriptive modeling. The three states are identified in Figure 7.9, annotated for mouse interaction.

From left to right in Figure 7.9, the states are Out of Range (State 0) for clutching or repositioning a mouse on a mouse pad, Tracking (State 1) for moving a tracking symbol (e.g., a cursor) on a display, and Dragging (State 2) for moving an icon on the display or for grouping a set of objects or a range of text. The model seems simple and obvious, and we might question its ability to add insight to the existing body of pointing device research. Yet the model can be extended to capture additional aspects of pointing device interaction such as multi-button interaction, stylus or finger input, and direct versus indirect input. See Buxton (1990) for further details.

As further evidence of the utility of Buxton's work, MacKenzie and Oniszczak (1998) used the three-state model to help characterize an interaction technique that didn't exist at the time Buxton's model was introduced. The insight led to a redesign

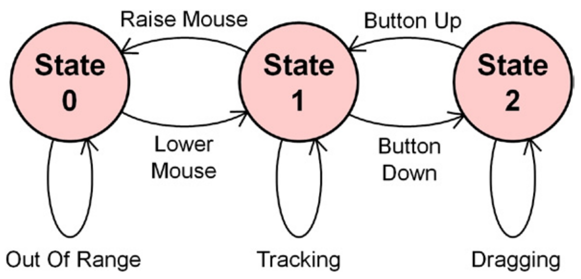


FIGURE 7.9

Buxton's three-state model for graphical input.

(Adapted from Buxton, 1990)

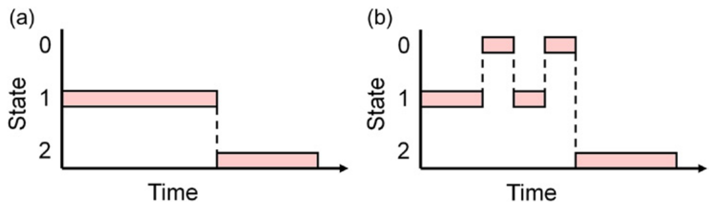


FIGURE 7.10

State transitions for dragging tasks: (a) Mouse. (b) Lift-and-tap touchpad. Dragging begins upon entering State 2.

(Adapted from MacKenzie and Oniszczak, 1998)

of an interaction technique that later appeared in the Blackberry Storm (Arthur, 2008a) and Apple MacBook (Arthur, 2008b). This work is briefly recounted here.

Never shy of innovation, Apple took a bold step in 1994 by commercializing a new pointing device in its PowerBook 500 notebook computer: the Trackpad touchpad (MacNeill and Blickenstorfer, 1996). Today, touchpads are the dominant pointing device for notebook computers, notepads, and other small mobile devices. One of the interaction techniques supported by touchpads is "lift-and-tap," where primitive operations like clicking, double-clicking, and dragging are implemented without a button. These new interaction primitives are easily represented by Buxton's three-state model. Figure 7.10 provides a simple comparison of the state transitions for dragging tasks (a) using a mouse and (b) using lift-and-tap on a touchpad.

Diagrams like this are evidence of the power of descriptive models such as Buxton's three-state model. Two observations follow: (1) lift-and-tap necessitates extra state transitions in comparison to a mouse, and (2) the use of state 1-0-1 transitions for lift-and-tap is confounded with clutching (not shown) which uses the same state transitions.[4] Among users' frustrations in using touchpads is that these

[4] *Clutching* refers to lifting the mouse to reposition it within its operating space.

primitives are difficult, awkward, or error prone. For example, if a touch following a lift is spatially displaced from the point of lifting, the system sometimes enters the Tracking state (State 1) instead of the Dragging state (State 2).

Armed with a deeper understanding of touchpad interaction, the state transitions on touchpads were redesigned. The additional pressure sensing capability of touchpads was used to implement state 1–2 transitions by "pressing harder."[5] A relay was added to provide both tactile and auditory feedback to inform the user of state transitions, much like the feedback in pressing a mouse button. Thus clicking, double clicking, and dragging were implemented on the touchpad, without a button, yet using the same state transitions as on a mouse. The complete details are presented in the 1998 proceedings of the ACM's SIGCHI (MacKenzie and Oniszczak, 1998).

Despite being introduced in 1990, Buxton's work continues to serve as a descriptive model for interactive systems. Contemporary applications include models for preview and undo (Appert, Chapuis, and Pietriga, 2012; Forlines, Shen, and Buxton, 2005), puck and stylus interactions for two-handed input (Hinckley, Czerwinski, and Sinclair, 1998), docking tasks for tabletop displays (Forlines, Wigdor, Shein, and Balakrishnan, 2007), camera control for spatial and temporal navigation of animated scenes (Burtnyk, Khan, Fitzmaurice, Balakrishnan, and Kurtenbach, 2002), modeling multi-touch on touchscreens (Wigdor et al., 2009), modeling panning and zooming on touchscreens (Malacria, Lecolinet, and Guiard, 2010), modeling the selection of moving targets (Al Hajri, Fels, Miller, and Ilich, 2011), and describing the rotation mode of a three DOF mouse (Almeida and Cubaud, 2007). In the latter case, an extra state was added ("State 3") to represent rotation.

Let's move across the continuum of the modeling space to models that work with closed-form mathematical equations—predictive models.

7.2 Predictive models

A predictive model is an equation. The equation predicts the outcome of a variable based on the value of one or more other variables (predictors). The outcome variable is a dependent variable, typically the time or speed in doing a task. It could also be accuracy, represented as spatial variability, error rate, or any other measure of human behavior. The only requirement is that the variable use continuous, or ratio-scale, data.

Most statistics sources call the predictor variable the *independent variable*. While correct, the terminology is problematic here. In experimental research, independent variable has a special meaning: it is a circumstance or characteristic that is manipulated. The vast majority of independent variables are nominal-scale attributes (e.g., device, feedback modality, display type, gender). This poses a problem, since a nominal-scale variable cannot serve as a predictor in a prediction equation

[5]The pressure sensing capability of a touchpad exists within the firmware as a separate mode of operation. In this mode, the device provides absolute *x*, absolute *y*, and *z*-axis data. The *z*-axis data corresponds to finger pressure on the touchpad surface.

(e.g., $3 \times device = ?$). Of course, independent variables in experimental research can also be ratio-scale attributes. Such variables can serve as predictors in prediction equations. Examples include distance to target, size of target, number of targets, angle of movement, system lag, joystick force, angle of tilt, decibel (dB) level of background noise, word size, number of choices, and so on. A predictor variable can also be a ratio-scale attribute of users such as age, years of computer experience, number of e-mails received per day, hours per day playing video games, geographic distance from a friend, and so on.[6]

Predictive models are useful in HCI. Like descriptive models, they allow a problem space to be explored. However, with a predictive model we are dealing with numbers, not concepts. In 1978 Card, English, and Burr (1978) presented what is likely the first predictive model in HCI. They conducted an experiment comparing the effect of four input devices (mouse, joystick, text keys, step keys) on users' speed and accuracy in selecting text on a CRT display. In many respects their work is straightforward; the methodology is as expected for an experiment with human participants. However, Card et al. went beyond a typical user study. Here are the first two sentences in the Discussion:

> *While these empirical results are of direct use in selecting a pointing device, it would obviously be of greater benefit if a theoretical account of the results could be made. For one thing, the need for some experiments might be obviated; for another, ways of improving pointing performance might be suggested. (p. 608)*

This is an inspired preamble to their discussion on building models—models of interaction that (a) embed a theoretical account of the underlying human processes and (b) serve as prediction tools for follow-up analyses of design alternatives. The remainder of their paper is about modeling using Fitts' law. They built and compared Fitts' law models for the mouse and joystick. Many dozens of HCI papers on Fitts' law have followed in the same vein. We will visit Fitts' law shortly, but first let's examine how a prediction equation is built.

7.2.1 Linear regression model

The basic prediction equation expresses a linear relationship between an independent variable (x, a predictor variable) and a dependent variable (y, a criterion variable or human response)

$$y = mx + b \qquad (1)$$

where m is the slope of the relationship and b is the y intercept. (See Figure 7.11.)

[6]"Geographic distance from a friend" is an excellent example of the diverse and multi-disciplinary nature of HCI research. This ratio-scale circumstance was used both as an independent variable and as a predictor in a prediction equation for research on maintaining online friendships when people move apart (Shklovski, Kraut, and Cummings, 2006).

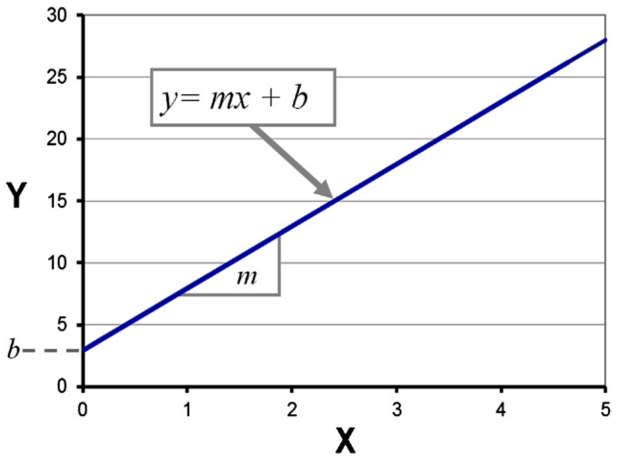


FIGURE 7.11

Linear relationship between an independent variable (*x*) and a dependent variable (*y*).

To build this equation, we first need a set of *x-y* sample points. Although any two ratio-scale variables will do, most commonly the *x-y* points combine the setting on an independent ratio-scale variable (*x*) with the measured value of a human response on a dependent variable (*y*). Since we are dealing with humans, variability is unavoidable, so the sample points are unlikely to lie on the line. They will scatter about. If the model is good, the points will be reasonably close to a straight line. How close is a key question.

Finding the best-fitting straight line involves a process known in statistics as *linear regression*. The objective is to find coefficients *m* and *b* in Equation 1 for the line than minimizes the squared distances (*least squares*) of the points from the line.[7] The result is the prediction equation—an equation that gives the best estimate of *y* in terms of *x*. Of course, the model is based on the sample points used in building the equation. If the data are good and the model is correct, a good prediction equation should emerge. There is also a built-in assumption that the relationship is linear, which is not necessarily the case. Let's visit an example in HCI.

As part of an experiment investigating text entry using a stylus on soft keyboards, MacKenzie and Zhang (2001) also wondered whether entry speed by stylus-tapping could be predicted from user's speed in touch-typing with a standard keyboard. The experiment involved 12 participants. The participants were given a pre-test to measure their touch-typing speeds. During the experiment, participants entered text using a stylus and a Qwerty soft keyboard displayed on an LCD tablet and digitizer. The pre-test touch-typing speed (independent variable) and

[7]Enter "linear regression" or "least squares" into Google or Wikipedia, and you will find all the details.

(a)

Participant	Stylus Tapping Speed (wpm)	Touch Typing Speed (wpm)
P1	18.2	42
P2	23.6	44
P3	26.0	32
P4	20.3	50
P5	20.3	36
P6	17.1	33
P7	24.0	74
P8	14.7	22
P9	20.3	31
P10	19.7	33
P11	22.4	25
P12	13.1	19

(b)

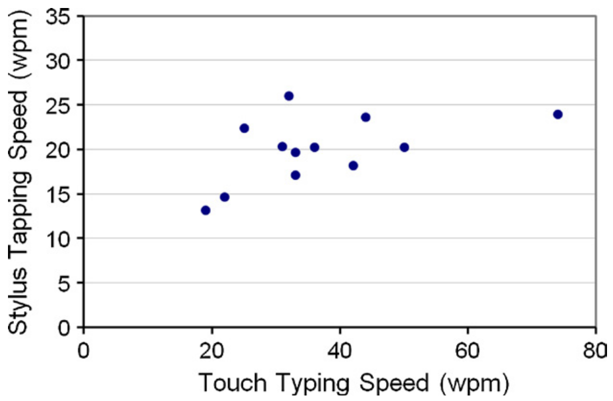


FIGURE 7.12

Relationship between stylus-tapping speed and touch-typing speed: (a) Data. (b) Scatter plot.

(Adapted from MacKenzie and Zhang, 2001, Fig. 4)

experimentally measured stylus-tapping speed (dependent variable) are given in Figure 7.12a for each participant.

Often as a precursor to building a prediction equation, a simpler question is posed: is there a relationship between the two variables? For this example, the question is this: do fast touch typists tend to be fast at stylus tapping?[8] Visualizing the data as a scatter plot helps (see Figure 7.12b). Yes, there seems to be a relationship. For example the slowest touch typist, P12 at 19 wpm, was also rather slow at stylus tapping (13.1 wpm). For the 12 points in the figure, the coefficient of correlation is $r = .5228$.[9] That's a modest positive correlation.

[8]The motivation is that proficient typists have a good visual image of letter positions on a Qwerty keyboard and, therefore, may require less visual scan time when using a stylus on a soft keyboard. Perhaps this phenomenon only applies to hunt-and-peck typists—typists who visually attend to the keyboard when typing. True touch typists tend to use muscle memory and do not visually attend to their keyboard. Perhaps they will exhibit a reverse phenomenon. More research is needed!

[9]The coefficient of correlation is computed in Microsoft Excel using the CORREL function.

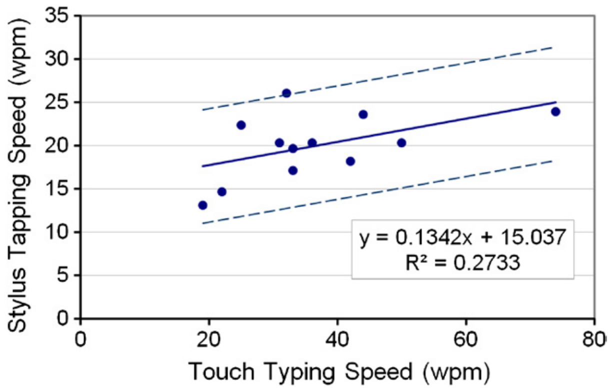


FIGURE 7.13

Scatter plot from Figure 7.12b embellished with linear regression line, prediction equation, squared coefficient of correlation (R^2) and dashed lines showing the 95% confidence interval.

The next step is to build the prediction equation—the best-fitting straight-line equation predicting stylus-tapping speed (y) from touch-typing speed (x). This is easily done using a spreadsheet application such as Microsoft Excel.[10] For the data in Figure 7.12a, the prediction equation is:

$$y = 0.1342\,x + 15.037 \tag{2}$$

Figure 7.13 is an embellished version of the chart in Figure 7.12b. As well as the scatter of points, it shows the line for the prediction equation along with the equation and the squared coefficient of correlation, $R^2 = .2733$. (By convention, R^2 is set in uppercase, r in lowercase.) R^2 is interpreted as the amount of variation in the data that is explained by the model. It is commonly articulated as a percent. So the model in Equation 2 explains about 27 percent of the variation of the data in Figure 7.12a. That's not very high, so in this case, the model is, at best, a modest predictor of stylus-tapping speed from touch-typing speed.[11]

One of the benefits of a predictive model is the potential to predict the outcome on a value of the predictor never actually visited. Note in Figure 7.12a that no participant had a touch-typing speed in the range of 60 wpm. Even so, we might

[10]First, the points are plotted as a "(XY) scatter" chart. With the points selected, the "Add Trendline" option from the Chart menu is used to add a linear regression line (trendline) to the chart. Options are available to include the equation and the squared correlation (R^2). If the goal is just to compute the slope (m) and intercept (b), then the SLOPE and INTERCEPT functions may be used.

[11]Low values of R^2 are common in the literature. Yin and Zhai (2006) report a linear regression model with $R^2 = .013$. As they noted, "the model accounted for very little of the variance in the actual selection (1.3%)."

conjecture that a user with a touch-typing speed of 60 wpm would have a stylus-tapping speed of:

$$y = 0.1342(60) + 15.037 = 23.1\,wpm \tag{3}$$

Given the scatter of points in Figure 7.12a, there is clearly some uncertainty surrounding this prediction. The standard error of estimate (*SE*) is a useful statistic for gauging this uncertainty. *SE* establishes confidence intervals around a prediction. For the data in Figure 7.12a, $SE = 3.39$ wpm.[12] Values within −1.96 *SE* and +1.96 *SE* of a prediction are within a 95% confidence interval. So for the model developed here, there is 95% confidence that a user whose touch-typing speed is 60 wpm will have a stylus-tapping speed between $23.1 - (1.96 \times 3.39) = 16.4$ wpm and $23.1 + (1.96 \times 3.39) = 29.7$ wpm. Dashed lines showing the 95% confidence window are included in Figure 7.13.

Usually linear regression models are reported giving the equation and R^2. Confidence intervals and *SE* are generally not given, although there are exceptions (Chung and Hossain, 2008; Johnsen, Raij, Stevens, Lind, and Lok, 2007; MacKenzie and Buxton, 1994). Sometimes the standard error is given separately for the slope and intercept coefficients in a linear regression model (Accot and Zhai, 2001; Cao et al., 2008; Pastel, 2006). Let's move on to a popular prediction model in HCI, Fitts' law.

7.2.2 Fitts' law

One of the most widely used models in HCI is Fitts' law. If an interaction involves a rapid-aimed movement, such as moving a finger or cursor to a target and selecting the target, there's a good chance someone has experimented with the interaction and used Fitts' law to model it. Fitts' law has three uses in HCI: (1) to learn if a device or interaction technique conforms to the model by building the prediction equation and examining the correlation for "goodness of fit," (2) to use the prediction equation in analyzing design alternatives, or (3) to use Fitts' *index of performance* (now *throughput*) as a dependent variable in a comparative evaluation. We'll examine each of these in the discussions that follow.

Since Fitts' law comes to HCI by way of basic research in experimental psychology, I'll begin with some background. Detailed reviews are provided elsewhere (MacKenzie, 1992; Meyer, Smith, Kornblum, Abrams, and Wright, 1990; Soukoreff and MacKenzie, 2004; Welford, 1968).

7.2.2.1 Background

Fitts was an experimental psychologist interested in applying information theory to human behavior. This was a common theme of research in the 1950s as it merged the idea of human performance with the contemporary and emerging mathematical concept of "information" in electronic communications. Fitts argued that the amplitude of an aimed movement was analogous to the information in an electronic

[12]The standard error of estimate (*SE*) is calculated in Microsoft Excel using the STEYX function. The formula assumes the data are from a sample, rather than the population.

signal and that the spatial accuracy of the move was analogous to electronic *noise*. Furthermore, he proposed that the human motor system is like a communications channel, where movements are like the transmission of signals. Fitts' analogy is based on Shannon's Theorem 17, expressing the information capacity C (in bits/s) of a communications channel of bandwidth B (in s^{-1} or Hz) as

$$C = B\log_2\left(\frac{S}{N} + 1\right) \tag{4}$$

where S is the signal power and N is the noise power (Shannon and Weaver, 1949, pp. 100–103).

Fitts presented his analogy—now his "law"—in two highly celebrated papers, one in 1954 (Fitts, 1954) and the second in 1964 (Fitts and Peterson, 1964). The 1954 paper described a serial, or reciprocal, target acquisition task where participants alternately tap on targets of width W separated by amplitude A (see Figure 7.14a). The 1964 paper described a similar experiment using a discrete task, where

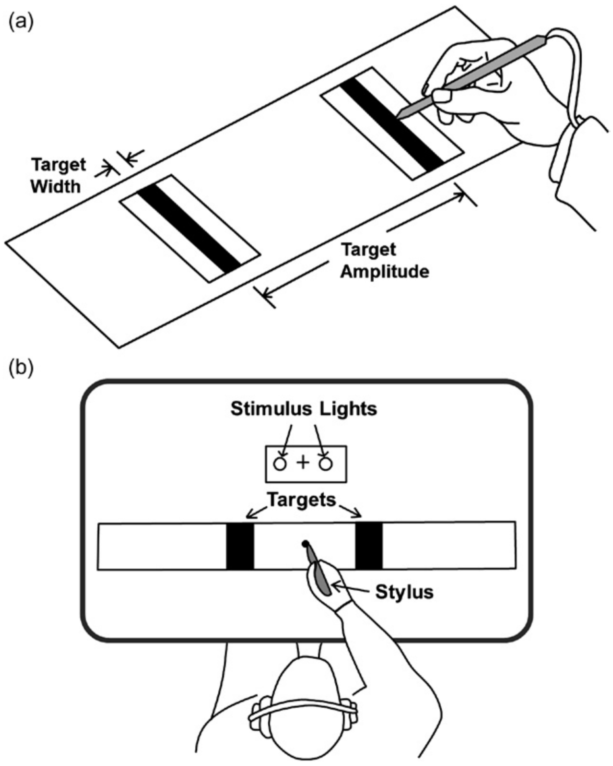


FIGURE 7.14

Experimental paradigm for Fitts' law: (a) Serial task. (b) Discrete task.

(Source: a, adapted from Fitts, 1954; b, adapted from Fitts and Peterson, 1964)

subjects selected one of two targets in response to a stimulus light (see Figure 7.14b). It is easy to imagine how to update Fitts' apparatus with computer input devices and targets rendered on a computer display.

The relationship Fitts examined is that between the amplitude of the movement task, the time taken, and the width, or tolerance, of the region within which the move terminates. In what may be the most understated conclusion in all of experimental psychology, Fitts summarized his findings as follows: "The results are sufficiently uniform to indicate that the hypothesized relation between speed, amplitude, and tolerance may be a general one." (Fitts, 1954, p. 389) Indeed, the robustness of Fitts' law is extraordinary. In the decades since Fitts' seminal work, the relationship has been verified countless times and in remarkably diverse settings, whether under a microscope or under water.

Fitts proposed to quantify a movement task's difficulty—*ID*, the *index of difficulty*—using information theory by the metric "bits." Specifically:

$$ID = \log_2\left(\frac{2A}{W}\right) \tag{5}$$

The amplitude (A) and width (W) in Equation 5 are analogous to Shannon's signal (S) and noise (N) in Equation 4. Note the offsetting influences of A and W in the equation. Doubling the distance to a target has the same effect as halving its size. An important thesis in Fitts' work was that the relationship between task difficulty and the movement time (*MT*) is linear. The following expression for *ID* was introduced later to improve the information-theoretic analogy (MacKenzie, 1992):

$$ID = \log_2\left(\frac{A}{W} + 1\right) \tag{6}$$

Because A and W are both measures of distance, the term in parentheses in Equation 6 is unitless. The unit "bits" emerges from the somewhat arbitrary choice of base 2 for the logarithm.

Fitts also proposed to quantify the human rate of information processing in aimed movements using *bits per second* as the units. This is a provocative idea, based purely on analogy, without a basis in human psychomotor behavior. Fitts' *index of performance*, now called *throughput* (*TP*, in bits/s), is calculated by dividing *ID* (bits) by the mean movement time, *MT* (seconds), computed over a block of trials:

$$TP = \frac{ID_e}{MT} \tag{7}$$

The subscript e in ID_e reflects a small but important adjustment, which Fitts endorsed in his 1964 paper. An adjustment for accuracy involves first computing the *effective target width* as

$$W_e = 4.133 \times SD_x \tag{8}$$

where SD_x is the standard deviation in a participant's selection coordinates.[13] Computed in this manner, W_e includes the spatial variability, or accuracy, in responses. In essence, it captures what a participant actually did, rather than what he or she was asked to do. This adjustment necessitates a similar adjustment to *ID*, yielding an "effective index of difficulty":

$$ID_e = \log_2\left(\frac{A}{W_e} + 1\right) \tag{9}$$

Calculated using the adjustment for accuracy, *TP* is a human performance measure that embeds both the speed and accuracy of responses. *TP* is most useful as a dependent variable in factorial experiments using pointing devices or pointing techniques as independent variables.

If using Fitts' law as a predictive model is the goal, then the movement time (*MT*) to complete a task is predicted using a simple linear equation:[14]

$$MT = a + b \times ID \tag{10}$$

The slope and intercept coefficients in the prediction equation are determined through empirical tests, typically using linear regression. The tests are undertaken in a controlled experiment using a group of participants and one or more input devices and task conditions.

7.2.2.2 Example

In this section I demonstrate how to build a Fitts' law predictive model, using a subset of the data from an experiment comparing pointing devices (MacKenzie and Jusoh, 2001). One device was an Interlink Electronics *RemotePoint*. The RemotePoint is a cordless pointing device operated in the air. Cursor position is controlled by a thumb-activated isometric joystick. Selection uses a trigger-like switch operated with the index finger. See Figure 7.15a. A Microsoft *Mouse 2.0* served as a baseline condition. Twelve participants performed a series of reciprocal point-select tasks across nine target conditions: $A = 40$, 80, 160 pixels crossed with $W = 10$, 20, 40 pixels. Using Equation 6, the target conditions span a range of difficulties from $ID = \log_2(40 / 40 + 1) = 1.00$ bits to $ID = \log_2(160 / 10 + 1) = 4.09$ bits. See Figure 7.15b.

The data for the mouse and RemotePoint conditions are given in Figure 7.16. There are nine rows, one for each *A*-*W* target condition. The three columns on the left are the target conditions (*A*, *W*, and the calculated index of task difficulty, *ID*). The remaining columns are based on the participants' responses. W_e is the *effective target width* (Equation 8). ID_e is the effective index of difficulty

[13] The effective target width (W_e) adjusts *W* to reflect a 4% error rate. For the full details and rationale, see Fitts (1964), Welford (1968, 145–149), MacKenzie (1992), or Soukoreff and MacKenzie (2004). For an argument against the adjustment for accuracy, see Zhai, Kong, and Ren (2004).

[14] It is important to distinguish between *building* a Fitts' law model and *using* a Fitts' law model. In building the model, ID_e should be used (so the model reflects both the speed and accuracy of participants' responses). In using the model, *ID* is used. The predictions will carry a 4% error probability.

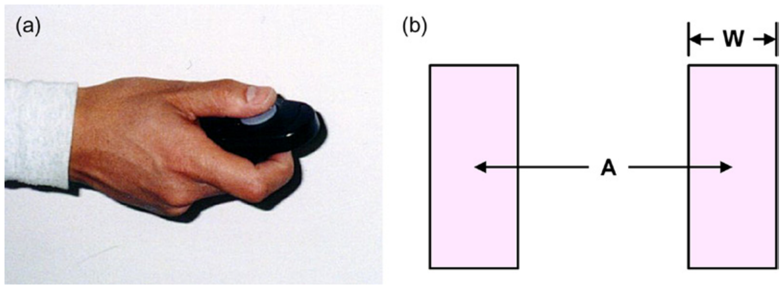


FIGURE 7.15

(a) Interlink Electronics RemotePoint. (b) Experiment task.

A (pixels)	*W* (pixels)	*ID* (bits)	Mouse W_e (pixels)	Mouse ID_e (bits)	Mouse *MT* (ms)	Mouse *TP* (bits/s)	RemotePoint W_e (pixels)	RemotePoint ID_e (bits)	RemotePoint *MT* (ms)	RemotePoint *TP* (bits/s)
40	10	2.32	11.23	2.19	665	3.29	13.59	1.98	1587	1.25
40	20	1.58	19.46	1.61	501	3.21	21.66	1.51	1293	1.17
40	40	1.00	40.20	1.00	361	2.76	37.92	1.04	1001	1.04
80	10	3.17	10.28	3.13	762	4.11	10.08	3.16	1874	1.69
80	20	2.32	18.72	2.40	604	3.97	25.21	2.06	1442	1.43
80	40	1.58	35.67	1.70	481	3.53	37.75	1.64	1175	1.40
160	10	4.09	10.71	3.99	979	4.08	10.33	4.04	2353	1.72
160	20	3.17	21.04	3.11	823	3.77	19.09	3.23	1788	1.81
160	40	2.32	41.96	2.27	615	3.69	35.97	2.45	1480	1.65
		Mean	23.25	2.38	644	3.60	23.51	2.35	1555	1.46

FIGURE 7.16

Experiment conditions (*A*, *W*, *ID*) and participants' responses (W_e, ID_e, *MT*, *TP*) for the standard mouse and the RemotePoint.

(Equation 9). *MT* is the mean movement time in milliseconds. *TP* is the throughput in bits/s (Equation 7).

A Fitts' law model can be built for both device conditions in Figure 7.16. The prediction equations use *MT* as the dependent variable and either *ID* or ID_e as the predictor variable. The main argument for using ID_e is that the model includes both the speed and accuracy of responses and therefore more fully encompasses participant behavior. The easiest way to build and demonstrate the models is to use a spreadsheet application. Figure 7.17 provides a visualization of the data as a scatter plot for each device. The chart also includes the regression lines, the prediction equations, and R^2. Both R^2 values are very high. The mouse model, for example, explains 97 percent of the variance in the observations.

The format of the prediction equations in Figure 7.17 can be improved. For the RemotePoint, the predicted time to move a cursor over a distance *A* to select a target of width *W* is:

$$MT = 597 + 408 \times \log_2\left(\frac{A}{W} + 1\right)\text{ms} \tag{11}$$

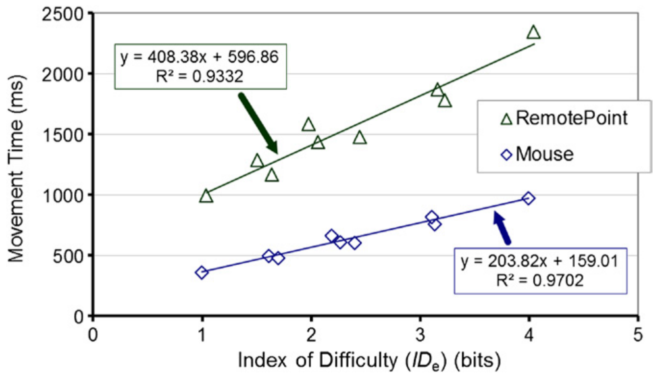


FIGURE 7.17

Scatter plot and regression lines for data in Figure 7.16b.

(Adapted from MacKenzie and Jusoh, 2001)

For the mouse, the prediction equation is:

$$MT = 159 + 204 \times \log_2\left(\frac{A}{W} + 1\right) \text{ ms} \tag{12}$$

What insight is found in the Fitts' law models in Figure 7.17? An initial observation is that point-select operations with both devices conform to Fitts' law (as is evident by the high values of R^2). This is the first use of Fitts' law, noted earlier. Also, it is apparent that the RemotePoint performed poorly compared to the mouse. The movement time was substantially longer for the RemotePoint, for all target conditions. The slopes of the regression lines demonstrate a substantially lower rate of information processing for the RemotePoint. The slope has units of ms/bit. The slope reciprocal, with a conversion to seconds, has units of bits/s. Thus a lower slope is a higher rate of information processing (throughput). The mouse is clearly a better device.[15] As an overall measure, the preferred calculation for throughput uses a division of means (Equation 7), as proposed by Fitts (1954). These values are along the bottom row in Figure 7.16. The throughput for the RemotePoint (1.46 bits/s) is less than half that for the mouse (3.60 bits/s).

There are many dozens of Fitts' law prediction models in the HCI literature. Some recent and representative examples are provided by Sällnas and Zhai (2003, Fig. 4), Wobbrock and Gajos (2007, Fig. 8), Po, Fisher, and Booth (2004, Fig. 2), Vertegaal (2008, Fig. 2), Rohs and Oulasvirta (2008, Fig. 11), Dixon, Guimbretière, and Chen (2008, Fig. 6), Perry and Hourcade (2008, Fig. 5), Baudisch, Cutrell, Hinckley, and Eversole (2005, Fig. 13), and Sproague, Po, and Booth (2006, Fig. 7).

[15] *Better*, here, refers to performance measured using task completion time, or Fitts' throughput. If the interaction involves standing in front of an audience, the *RemotePoint* is likely a better choice.

There is also some debate on the method of calculating throughput (see Soukoreff and MacKenzie, 2004; Zhai et al., 2004).

In the 1950s, when Fitts proposed his model of human movement, graphical user interfaces and computer pointing devices didn't exist. Yet throughout the history of HCI (since Card et al., 1978), research on computer pointing is inseparable from Fitts' law. The initial studies focused on device comparisons and model conformity. As the field evolved and matured, however, Fitts' law found its way into diverse topics—topics only peripherally related to pointing devices. Examples include expanding targets, hidden targets, fish-eye targets, crossing-based interfaces, path following, steering, pointing on the move, eye tracking, force feedback, gravity wells, multi-monitor displays, magic lenses, and so on. Research in these topics, and more, has thrived on the theory and predictive modeling techniques inspired and guided by Fitts' law. This is Fitts' legacy to research in human-computer interaction.

In looking back, the success of Fitts' law in HCI is due in no small measure to the endorsement the model received early on from Card, Moran, and Newell in *The Psychology of Human-Computer Interaction* (1983). Their model human processor (MHP) included two general psychological information theoretic models as guiding principles for the field (1983, 26–27). One was Fitts' law. The other was the Hick-Hyman law for choice reaction time.

7.2.3 Choice reaction time

The Hick-Hyman law for choice reaction time, like Fitts' law, arrived in HCI by way of basic research in experimental psychology—research seeking to model human behavior according to information processing principles (Hick, 1952; Hyman, 1953). The model also takes the form of a prediction equation. Given n stimuli, associated one-for-one with n responses, the time to react (RT) to the onset of a stimulus and choose the correct response is given by

$$RT = a + b\log_2(n+1) \tag{13}$$

where a and b are empirically determined constants. Reasonable values for the constants are $a \approx 200\,\text{ms}$ and $b \approx 150\,\text{ms/bit}$ (Card et al., 1983, p. 27, p. 76).[16] Figure 7.18 shows a typical setup for a choice reaction time experiment. At random intervals, a stimulus light is activated and the human responds by pressing the associated key as quickly as possible.

As with Fitts' law, the log term is analogous to the information content of the task and has units of "bits." There is an interesting twist to this for choice reaction time. If some of the choices are more probable than others, the information content of the task is reduced. This in turn reduces the choice reaction time. Let's explore this idea.

[16]The model is sometimes expressed as $RT = a + b\log_2(n)$ and sometimes without an intercept. See Card, Moran, and Newell (1983, 71–76) and Welford (1968, 61–70) for detailed discussions.

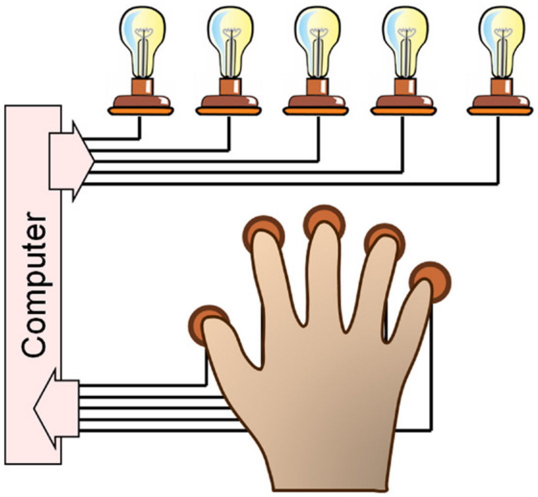


FIGURE 7.18

Paradigm for choice reaction time experiments.

For a set of alternatives with different probabilities, the information (H) is

$$H = \sum_i p_i \log_2 \left(\frac{1}{p_i} + 1 \right) \tag{14}$$

where p_i is the probability of occurrence of the i^{th} item in the set. Consider a choice selection task where the choice is among 26 alternatives. If the alternatives appear with equal probability, the information content of the task is simply:

$$H = \log_2(27) = 4.75\,\text{bits} \tag{15}$$

If the alternatives appear with different probabilities, the information is reduced according to Equation 14. This occurs, for example, if the 26 stimuli are letters of the English alphabet and they appear with the same probability as in English writing. An example set of letter frequencies and probabilities for English is shown in Figure 7.19. The data were obtained from the British National Corpus[17] using the word-frequency list of Silfverberg et al. (2000).

Since *e* appears with greater frequency than, for example, *z*, the task is slightly easier because there is a small opportunity to anticipate the choices at the onset of a stimulus. In so doing, the reaction time is reduced. Overall, there is less uncertainty, or information, in the task. The information content of the task is reduced from 4.75 bits (Equation 15) to 4.25 bits (Equation 14, illustrated in Figure 7.19).

The Hick-Hyman law has surfaced in a few contexts in HCI research. Card et al. (1983, 74) describe an example of a telephone operator selecting among ten buttons

[17] ftp://ftp.itri.bton.ac.uk

Letter	Frequency	Probability (p)	$p \log_2(1/p + 1)$
a	24373121	0.0810	0.3028
b	4762938	0.0158	0.0950
c	8982417	0.0299	0.1525
d	10805580	0.0359	0.1742
e	37907119	0.1260	0.3981
f	7486889	0.0249	0.1335
g	5143059	0.0171	0.1008
h	18058207	0.0600	0.2486
i	21820970	0.0725	0.2819
j	474021	0.0016	0.0147
k	1720909	0.0057	0.0427
l	11730498	0.0390	0.1846
m	7391366	0.0246	0.1322
n	21402466	0.0711	0.2783
o	23215532	0.0772	0.2935
p	5719422	0.0190	0.1092
q	297237	0.0010	0.0099
r	17897352	0.0595	0.2471
s	19059775	0.0633	0.2578
t	28691274	0.0954	0.3358
u	8022379	0.0267	0.1404
v	2835696	0.0094	0.0636
w	6505294	0.0216	0.1203
x	562732	0.0019	0.0170
y	5910495	0.0196	0.1119
z	93172	0.0003	0.0036
		H =	4.25

FIGURE 7.19

Letters of the English alphabet, with frequencies and probabilities. Information (*H*) is calculated using Equation 14.

when the light behind a button comes on. Landauer and Nachbar (1985) applied the Hick-Hyman law in measuring and predicting the time to select items in hierarchical menus. Besides confirming the suitability of the law to this category of task, they also found empirical support that breadth should be favored over depth in hierarchical menus. Note that choice reaction is different from visual search, which is a linear function of the number of items. Landauer and Nachbar eliminated visual scanning in the task by ensuring that "the sets of choice alternatives were well practiced and well-ordered so that it was not necessary to search them sequentially to find the target location" (Landauer and Nachbar, 1985, p. 73).

Ruiz et al. (2008) used the Hick-Hyman law to model the perception, planning, and activation time for users to switch modes with their non-dominant hands in a tablet interface. Despite these efforts and a few others, the Hick-Hyman law has failed to gain the same momentum in HCI as Fitts' law (Seow, 2005). One reason is that the model alone is of limited use. In most situations there are additional behaviors coincident with choice reaction, such as movement or visual search and these complicate applying the model.

7.2.4 The keystroke-level model

One of the earliest and certainly one of the most comprehensive predictive models in the HCI literature is the keystroke-level model (KLM) by Card, Moran, and Newell (1980; 1983, ch. 8). Unlike Fitts' law and the Hick-Hyman law, the KLM was developed specifically for analyzing human performance in interactive computing systems. It was offered as a practical design tool, or as the authors called it, an *engineering model*. The model predicts expert error-free task completion times using the following elements:

- Task (or series of sub-tasks)
- Method used
- Command language of the system
- Motor skill parameters of the user
- Response time parameters of the system

The model is useful in situations where the sequence of interactions in performing a task is known. For example, if the task is "delete a file," the KLM can predict the time to do the task provided the interactions (operators) can be explicitly specified. If there are two or three different methods to do the task (e.g., mouse + menu selection versus keyboard + command entry), then the KLM can predict the time for each method. If used at the design stage, then design alternatives may be considered and compared.

The KLM is not useful if the details of the interactions are not known. A related model, GOMS (Goals, Operators, Methods, Selection rules), operates at a higher level and attempts, among other things, to predict the method the user will adopt (Card et al., 1983, ch. 5). This is complicated, as it builds upon the cognitive information processing activities of the user and also assumes that users act rationally in attaining their goals.

A KLM prediction requires a task to be broken down into a series of subtasks, or primitive operations. The total predicted time is the sum of the subtask times. The model works with four motor-control operators (K = keystroking, P = pointing, H = homing, D = drawing), one mental operator (M), and one system response operator (R):

$$t_{\mathrm{EXECUTE}} = t_{\mathrm{K}} + t_{\mathrm{P}} + t_{\mathrm{H}} + t_{\mathrm{D}} + t_{\mathrm{M}} + t_{\mathrm{R}} \qquad (16)$$

Some operations are omitted or repeated, depending on the task. For example, if a keying subtask requires n keystrokes, t_{K} becomes $n \times t_{\mathrm{K}}$. Each t_{K} operation is assigned a value according to the skill of the user, with values ranging from $t_{\mathrm{K}} = 0.08\,\mathrm{s}$ for highly skilled typists to $t_{\mathrm{K}} = 1.20\,\mathrm{s}$ for a novice working with an unfamiliar keyboard. The pointing operator, t_{P} , is a constant based on Fitts' law. The operators and their values from the original KLM are given in Figure 7.20.

7.2.4.1 Original KLM experiment

To validate the KLM, Card, Moran, and Newell conducted an experiment using 14 tasks performed using various methods. Task T1, for example, was "Replace one

Operator	Description	Time (s)
K	PRESS A KEY OR BUTTON Pressing a modifier key (e.g., shift) counts as a separate operation, Time varies with typing skill:	
	Best typist (135 wpm)	0.08
	Good typist (90 wpm)	0.12
	Average skilled typist (55 wpm)	0.20
	Average non-secretary typist (40 wpm)	0.28
	Typing random letters	0.50
	Typing complex codes	0.75
	Worst typist (unfamiliar with keyboard)	1.20
P	POINT WITH A MOUSE Empirical value based on Fitts' law. Range from 0.8 to 1.5 seconds. Operator does *not* include the button click at the end of a pointing operation	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. Drawing with the mouse constrained to a grid.	.9 n_D + .16 l_D
M	MENTALLY PREPARE	1.35
R(*t*)	RESPONSE BY SYSTEM Different commands require different response times. Counted only if the user must wait.	*t*

FIGURE 7.20

Keystroke-level model (KLM) operators and values.

(From Card et al., 1983, 264)

5-letter word with another (one line from previous task)." It was performed on three different systems, each with a unique command set. On one system, POET, the required sequence of subtasks was as follows:

Jump to next line	**M K**[LINEFEED]
Issue Substitute command	**M K**[S]
Type new word	5**K**[word]
Terminate new word	**M K**[RETURN]
Type old word	5**K**[word]
Terminate old word	**M K**[RETURN]
Terminate command	**K**[RETURN]

The operators for each subtask are shown on the right. The task required four mental operations (M) and 15 keystroking operations (K):

$$t_{\text{EXECUTE}} = 4 \times t_M + 15 \times t_K \qquad (17)$$

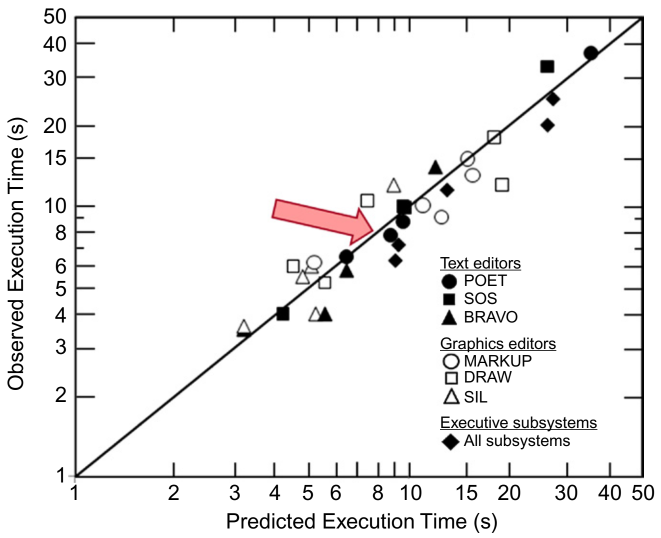


FIGURE 7.21

Observed versus predicted execution times for tasks in the KLM validation experiment. The arrow shows task T1 with POET.

(Adapted from Card et al., 1983, 277)

t_M was set to 1.35 s (Figure 7.20). t_K was set to 0.23 s, based on the mean keystroking time of the participants, determined in a five-minute pre-test. The predicted execution time for task T1 on POET was therefore

$$t_{EXECUTE} = 4 \times 1.35 + 15 \times 0.23 = 8.85\,\text{s} \tag{18}$$

In the experiment, twelve users performed the task an average of 27 times each. The mean execution time was 7.8 s ($SE = 0.9$ s). So the observation deviated from the prediction by ≈11%. The results were similar for the other tasks. Figure 7.21 shows the observed versus predicted task execution times for all 14 tasks (32 tasks, counting method variations). The coalescing of points about the diagonal is clear. This provides a general validation of the model. An arrow identifies the result for task T1 discussed above.

A model is a malleable tool that researchers can and should mold in any way that allows the model to tease out issues or opportunities in a problem space—opportunities to build a better interface or better interaction technique. Card, Moran, and Newell taught this in their *parametric analysis* or *sensitivity analysis* of the KLM. A predictive model is built on parameters. If parameters become variables and slide back and forth, what is the effect on the model's outcome? How sensitive are the predictions to changes in the parameters? This was explored in a series of arguments and demonstrations (1983, 287–293). Figure 7.22 is an example for an editing task. The figure charts $t_{execute}$ for three interaction methods as

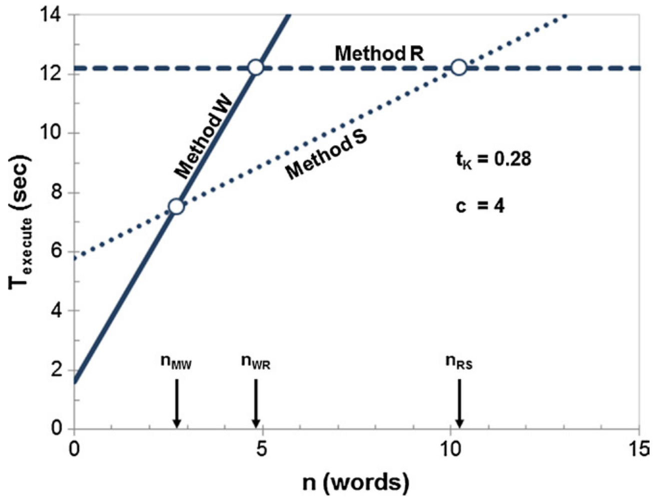


FIGURE 7.22

Parametric analysis showing the sensitivity of predictions for three methods based on changes in *n*—the location of an edit relative to the current location.

(Adapted from Card et al., 1983, 290)

a function of *n*—the distance in words from an initial location to the location of a misspelled word to correct. Viewing from left to right, optimal performance is achieved with Method W for $n < 2.7$, Method S for $2.7 < n < 10.2$, and Method R for $n > 10.2$. If the designer is choosing among candidate methods, and it is known that certain values of *n* are common, then the parametric analysis in Figure 7.22 provides precisely the information needed. A design choice is made supported by evidence obtained from a model of the interaction. Similar sensitivity analyses have appeared since Card et al.'s original KLM publication (e.g., Accot and Zhai, 2001; Clawson, Lyons, Rudnick, Iannucci, and Starner, 2008; W. D. Gray, John, and Atwood, 1993).

7.2.4.2 Modern applications of the KLM

Early experiments with the KLM related primarily to text editing and document management. The tasks were things like "delete a line of text," "add a label to a box," or "transfer a file to another computer, renaming it" (Card et al., 1983, p. 272). These tasks are just as common today as they were in the early 1980s; however, the available task methods are quite different. Transferring a file to another computer is good example. Today this task is likely to involve a mix of mouse and keyboard operations with a GUI-based FTP (file transfer protocol) application. In the original experiment, the task involved FTP as well, but the method only involved a keyboard. The subtask coding included 31 keystroking operations and 0 pointing operations.

Considering that the KLM is more than 25 years old, it is not surprising that some of the operators need updating. The pointing operator (P), for example, is a constant of 1.10 seconds. With the ubiquity of the mouse in today's GUIs, this operator could benefit from the additional fidelity afforded with a Fitts' law prediction. Using the mouse model from Figure 7.17, for example, a reasonable substitute for P is

$$t_P = 0.159 + 0.204 \times \log_2\left(\frac{A}{W} + 1\right) \tag{19}$$

If there are numerous mouse operations in a task undergoing KLM predictions, then the added precision in Equation 19 will improve the analysis. For example, if a subtask involves clicking a 1.2 cm wide toolbar button, having just clicked a neighboring button 3.2 cm away, then the pointing time is[18]

$$t_P = 0.159 + 0.204 \times \log_2\left(\frac{3.2}{1.2} + 1\right) = 0.45\text{ s} \tag{20}$$

Alternatively, if the mouse pointer must move, say, 44.6 cm to reach the same toolbar button, the pointing time is

$$t_P = 0.159 + 0.204 \times \log_2\left(\frac{44.6}{1.2} + 1\right) = 1.22\text{ s} \tag{21}$$

The predictions above include the terminating mouse-button click to select the target, unlike the original P operator.

As an example of applying the KLM to current GUI interaction, consider the editing operations to change the font style and font family for text while word processing. Figure 7.23 shows the operations to change "M K" to boldface in the Arial font.

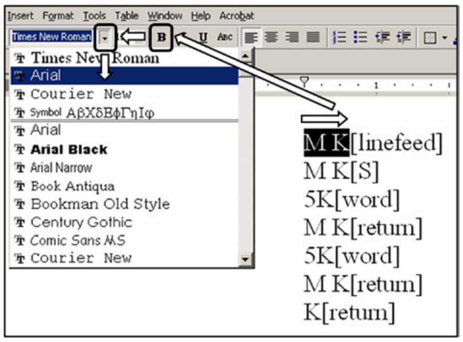


FIGURE 7.23

Mouse operations to change "M K" to boldface in the Arial font. The application is MS Word.

[18]For Fitts' law studies with the mouse, pointing times of 500 ms or less are common for easy tasks ($ID < 3$ bits). The minimum mouse pointing time in the original KLM was 800 ms (see Figure 7.20) which seems high.

The image is a screen snap for the editing performed above using Microsoft Word to characterize task T1. Four pointing operations are required: select the text, select Bold, select the drop-down arrow in the Font list, and select Arial.

A KLM coding of the subtasks is given in Figure 7.24.

Here, the pointing operator appears as P[A,W] to specify the movement amplitude and target width of the pointing task. The values shown were obtained using a ruler applied to Figure 7.23.[19] The execution time for each P operation is given in the right-hand column, computed using Equation 19. As there are four distinct subtasks, each is coded with a mental operator (M) as well as a pointing operator. The predicted task execution time is

$$t_{\mathrm{EXECUTE}} = 4 \times t_{\mathrm{M}} + \sum t_{\mathrm{P}} = 4 \times 1.35 + 2.71 = 8.11\,\mathrm{s} \tag{22}$$

As with the keystroking operator (K) in the original KLM, applications of the model using P[A, W] should use a pre-test to determine the coefficients in the Fitts' law model for the pointing device and method involved.

Of course, the same task can be done using the keyboard. A set of keyboard subtasks achieving the same effect is shown in Figure 7.25.

Here we see the same four mental operators; however, there is an assortment of keystroking actions, as per Microsoft Word. The KLM uses a separate keystroking

Mouse Subtasks	KLM Operators	t_P (s)
Drag across text to select "M K"	**M P**[2.5, 0.5]	0.686
Move pointer to Bold button and click	**M P**[13, 1]	0.936
Move pointer to Font drop-down button and click	**M P**[3.3, 1]	0.588
Move pointer down list to Arial and click	**M P**[2.2, 1]	0.501
	$\sum t_P =$	2.71

FIGURE 7.24

Mouse subtasks and KLM operators for editing operations in Figure 7.23.

Keyboard Subtasks	KLM Operators
Select text	**M P**[shift] 3**K**[→]
Convert to boldface	**M K**[ctrl] **K**[b]
Activate Format menu and enter Font sub-menu	**M K**[alt] **K**[o] **K**[f]
Type *a* ("Arial" appears at top of list)	**M K**[a]
Select "Arial"	**K**[↓] **K**(Enter)

FIGURE 7.25

Keyboard subtasks and KLM operators for the task in Figure 7.23.

[19] If the measurements are done on a display, they vary according to the display size and the zoom-factor of the document view. Since target amplitude and width appear as a ratio in Fitts' index of difficulty, scaling up or down does not affect the Fitts' law predictions.

value for pressing a modifier key, such as SHIFT. The predicted execution time using the keyboard is

$$t_{\mathrm{EXECUTE}} = 4 \times t_{\mathrm{M}} + 12 \times t_{\mathrm{K}} = 4 \times 1.35 + 12 \times 0.75 = 14.40\ \mathrm{s} \qquad (23)$$

This prediction uses the "typing complex codes" setting for t_{K} (0.75 s; see Figure 7.20). Depending on the user, this value might be too high or too low. Figure 7.26 explores this by showing the sensitivity of the prediction to changes in the keystroking time, t_{K}. Markers on the keyboard line show the t_{K} settings from Figure 7.20. Of course, changes in t_{K} do not affect the mouse prediction. The keyboard is slower than the mouse over all but the fastest three settings for t_{K}. Given that these settings correspond to keystroking times while typing at 55+ wpm, it is unlikely the keyboard method would be faster than the mouse method for any user, including experts.[20]

Of course, the mouse model deserves the same scrutiny as the keystroking operations. The mouse predictions are sensitive to the slope and intercept coefficients in the Fitts' law model. The first subtask was a dragging operation. Strictly speaking, a separate Fitts' law dragging model is required, since it is known that dragging operations are less efficient than pointing operations (MacKenzie, 1992).

The example above is a common task on today's GUI systems. Other tasks offer a great range of possibilities for keystroke-level modeling. Examples include scrolling a document with a mouse wheel, finger or thumb input on a mobile phone (perhaps while walking), two-thumb input on a mini-Qwerty keyboard, or stylus input on a PDA. How are these tasks modeled with a keystroke-level model? Clearly, there are many issues to consider in contemporary applications of the KLM. Despite this—or should I say, *motivated by this*—researchers continue

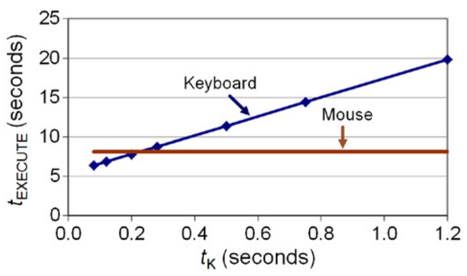


FIGURE 7.26

Sensitivity of prediction to keystroking time, t_{K}.

[20]The KLM applies to expert users *doing a task for the first time*. It is not intended for highly over-learned actions (Bonnie John, personal communication, January 2005). If a human operator were to repeatedly perform the task in this example, the task becomes over-learned. It could be modeled using the KLM; however, in such case, there are no mental operators and the keystroking time is substantially reduced.

to use the KLM to model interactions in novel HCI contexts. Examples of recent KLM applications include models for attention shifts with mobile phones (Holleis, Otto, Hussmann, and Schmidt, 2007), stylus-based circling gestures (Hinckley et al., 2006), managing folders and messages in e-mail applications (Bälter, 2000), predictive text entry on mobile phones (Ahmed and Stuerzlinger, 2010; M. Dunlop and Crossan, 2000; Y. Liu and Räihä, 2010), task switching in multi-monitor systems (Hashizume, Kurosu, and Kaneko, 2007), mode switching on tablet PCs (Ruiz et al., 2008), and distractions with in-vehicle information systems (IVIS) (Pettitt, Burnett, and Stevens, 2007). In other cases, the KLM is an embedded component of a comprehensive yet general-purpose modeling tool. An example is *CogTool-Explorer* (Teo, John, and Blackmon, 2012).

Much of the work just cited extends the KLM by adding or refining operators to suit the application. For example, Ruiz et al. (2008) defined a new operator t_{INT}, the interval between a mode switch with the non-dominant hand and the beginning of the next action. Pettitt et al. (2007) define t_{RF} for "reach far," the time to reach from a car's steering wheel to an IVIS (in-vehicle information system). As mentioned above, a model is a malleable tool. Use it, modify it, to suit the problem.

In all cases, use of the KLM entails breaking a task into subtasks and building an overall prediction using Equation 16, perhaps updated with new or refined operators. If the particular KLM implementation is known to work, then the predictions may be used in a true design or engineering setting—the original intention. However, if the model is being extended or applied in a novel setting as part of a research initiative, then validation is required. This is typically done in a formal or semi-formal experiment, where the task is performed with users while task completion times are observed and measured. The predicted and observed times are compared. If they are the same or similar, the KLM is validated. Deviations between predictions and observations signal a problem, but not necessarily a problem in the KLM. The discrepancy may indicate a problem in the task coding. The insight can be valuable. As Hinckley et al. (2006, 187) note:

> *[The discrepancy] shows where the techniques deviate from the model, indicating the presence of hidden costs. These costs might include increased reaction time resulting from planning what to do next, mental pauses, or delays while the user attends to visual feedback after performing an action. Our methodology cannot attribute these costs to a specific cause. It just lets us deduce that a hidden cost must exist in a specific portion of the task. This is sufficient to generate many insights as to where the bottlenecks to performance lie and what parts of a technique might be improved.*

So, as with descriptive models, predictive models like the KLM are tools for thinking, for generating insight.

Hinckley et al. refer to reacting, planning, pausing, and attending as hidden costs contributing to deviations between predictions and observations. These intangibles are problematic: It is difficult to get inside a user's mind to know what he or she is contemplating, let alone attribute a time cost to mental processes in a model.

Arguably, the mental operator (M) in the KLM is its Achilles heel. It is a single operator, pegged at 1.35 s, and is used for all mental processes. The human mind is simply too rich to be distilled so uniformly.

For this reason, researchers often do not model or predict execution time, but instead, model or predict only the keystrokes or other primitive actions. This makes for a much simpler problem. Actions, such as keystrokes, stylus taps, finger gestures, button clicks, gaze shifts, menu accesses, and so on, often capture important aspects of an interaction method apart from the human performance element. And modeling such actions is relatively straightforward. The end result is a model that is accurate but limited. While optimizing an "action parameter" is a noble cause, the connection to user performance is tenuous at best. Hidden costs are real costs, and the earlier they are accounted for—that is, at the modeling or design stage—the better.

7.2.4.3 The KLM and (Predictive) text entry

A classic HCI example of keystroke-level modeling, minus the hidden costs, is text entry, particularly mobile text entry. The mobile phone keyboard, being ambiguous, bears a special challenge for text entry. The relevant statistic for this is *KSPC* (keystrokes per character), defined as the number of keystrokes required, on average, for each character of text entered using a given input method in a given language (MacKenzie, 2002). It is known, for example, that a mobile phone has $KSPC \approx 2.023$ for multi-tap and $KSPC \approx 1.007$ for predictive text entry (*T9*). If word completion or word prediction is added, $KSPC < 1$. However, these figures do not account for the performance costs of visually attending to the interface. Let's consider how these costs might be accommodated in a keystroke-level model.

The original KLM experiment did not include a task such as "enter a 43-character phrase of text." The task was likely considered too trivial to bother with, and that's a reasonable position. The KLM prediction would reduce to $43 \times t_K$, where t_K is the keystroking time of the participant, based on a pre-test for typing speed. In essence, the task would just confirm the pre-test.

However, if the target system is a mobile phone and the text-entry method is multi-tap, the task seems appropriate for a KLM prediction. Or is it? Consider the 43-character phrase "the quick brown fox jumps over the lazy dog." Entering it using multi-tap requires 88 keystrokes.[21] In this case, the KLM prediction reduces to $88 \times t_K$, where t_K is the keystroking time of the participant, based on a pre-test. Of course, the pre-test is on a mobile phone using multi-tap. Once again, the task simply confirms the pre-test. It is important to remember that the KLM is a model for expert users. The participants would be experts with multi-tap on mobile phones.

However, interaction with a mobile phone keypad is distinctly different from two-handed typing on a computer keyboard. A mobile phone is typically held in the

[21]The words (in keystrokes) are as follows: the (84433S) quick (778844422255S) brown (22777666966S) fox (33366699S) jumps (58867N7777S) over (66688833777S) the (84433S) lazy (55529999N999S) dog (36664S). S is the space key. N is a *next* key (e.g., ↓) to segment consecutive letters on the same key.

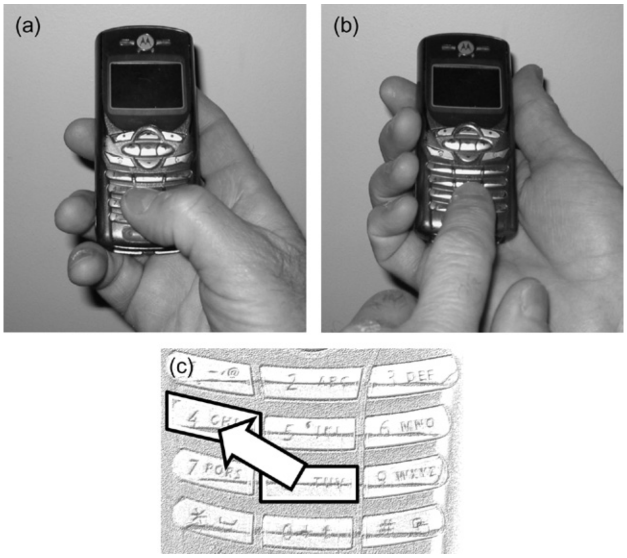


FIGURE 7.27

Interaction with a mobile phone keypad: (a) Thumb. (b) Index finger. (c) Fitts' law movement for "h" preceded by "t."

hand with keys pressed either by the thumb of the supporting hand or by the index finger of the opposite hand, as illustrated in Figure 7.27a and Figure 7.27b, respectively. Since only a single digit is used for input, the time for each key press can be modeled by Fitts' law, as shown in Figure 7.27c for pressing *h* preceded by *t*. Many multi-tap key presses are on the same key, but this too is a Fitts' law task (with $A = 0$; $MT \approx 160\,\text{ms}$). So a more refined prediction for multi-tap text entry can use the KLM pointing operator, with Fitts' law coefficients determined in a pre-test. Silfverberg et al. (2000) developed this approach in detail, presenting separate Fitts' law models for the thumb and index finger.

For touch typing on a Qwerty keyboard or multi-tapping on a mobile phone, the KLM's mental operator (M) is not needed. The user knows what to do and does it according to his or her keystroking expertise with the method. This is not the case for predictive text entry. Systems that involve word completion, word prediction, phrase prediction, or other language-based adaptive or predictive features are well known to tax the attention of users (e.g., Keele, 1973; Koester and Levine, 1994a, 1994b; Silfverberg et al., 2000). Predictive text entry techniques require the user to attend to and consider the system's behavior. Modeling this requires the KLM's mental operator (M) or something similar. Let's consider the possibilities.

Predictive text entry on a mobile phone (*T9*) requires only one keystroke per character as a word is entered. The system uses an internal dictionary to map candidate words to the key sequence. When the user finishes the keystrokes for a word,

(a)

2	a
23	be
233	bed
2337	beer
2337n	beds
2337nn	adds
2337nnn	bees
2337nnnn	beep
2337nnnn0	beep_

(b)

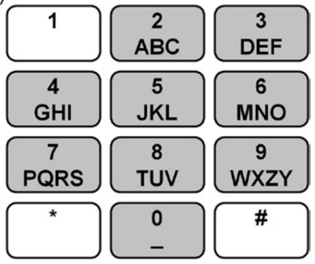


FIGURE 7.28

(a) Key sequence (*left*) and display progress (*right*) for input of *beep* using predictive text entry. Note: *n* is the *next* key (e.g., *). (b) Mobile phone keypad (for reference).

there is a good chance the desired word is displayed.[22] Unfortunately, "a good chance" isn't good enough; the uncertainty keeps the user on guard, monitoring the system's display. This creates an on-going attention demand ("Has my word appeared?"). Furthermore, input sometimes generates *collisions*—multiple words matching the key sequence. The words are presented in order of their probability in English and the user cycles through the possibilities to select the desired word. So there is an additional attention demand to navigate the list.

To further develop this idea, Figure 7.28a shows the key sequences (left) and display progress (right) for inputting "beep" on a mobile phone using predictive entry. For reference, a mobile phone keypad is shown in Figure 7.28b. The key sequence includes pressing *n* for *next* (e.g., $*$) to navigate the candidate list when collisions occur. After entering 2337, *beer* appears on the display. Four presses of next are required to navigate the list to reach the desired word. When the desired word appears, 0 is pressed to select it and append a space.

How are the interactions above modeled using the KLM? Clearly a raw breakdown using only the keystroking operator (K) or the pointing operator (P) is inappropriate. The interactions require mental processing as well as keying or pointing. The KLM's mental operator (M) is for mentally preparing to take the next action (Card et al., 1983, p. 263). This doesn't seem appropriate here, since the user must *perceive*, *decide*, and *react*. Yet a variant of M is clearly needed. Figure 7.29 proposes a task coding with nine keying operations and five mental operations. M is coded as M_P, for performing a *physical match* between a stimulus (the presented word) and a code stored in the user's short-term memory (the desired word). After keying 2337, the user sees the wrong word. The user reacts by pressing next (*n*). A similar M_P occurs after each press of next until the desired word appears, whereupon the user presses 0 to select the word and append space. An assumption here is that the candidate words are presented to the user one at a time, as is typical of most mobile phones.

One question to ask in applying KLM principles to predictive interfaces is: what is an expert user? The question is not as odd as one might think. Remember, even the

[22]For English, one estimate is that 95 percent of the words can be entered unambiguously (Silfverberg et al., 2000).

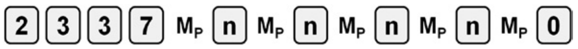


FIGURE 7.29

Key sequences and mental operators for the input of "beep" (see Figure 7.28).

Rank	Word	Keystrokes	Higher Ranking Colliding Word (rank)
47	if	43n0	he (15)
51	no	66n0	on (13)
63	then	8436n0	them (57)
72	me	63n0	of (2)
78	these	84373n0	there (35)
105	go	46n0	in (6)
118	us	87n0	up (56)
159	home	4663n0	good (115)
227	night	64448n0	might (141)
298	war	927n0	was (10)

FIGURE 7.30

Top ten ambiguous words requiring one press of next.

original KLM used a sliding scale for expertise. The keystroking operator (K) was tuned according to the typing skill of the user. Is there a similar sliding scale for the mental operator? Perhaps. Consider entering *the* (843), *of* (63), *and* (263), or other common words in English. Novices might monitor the display after entering 843 to confirm that *the* is displayed, but with a little practice, monitoring isn't necessary. 8430 produces *the*—no need to monitor the display! So the issue is not about tuning the value of M_P according to expertise, but in deciding when and where the mental operator is present.

The presence of, or need for, the M_P operator can be explored by examining a ranked list of English words and considering each word's position within ambiguous word sets. For this analysis, a ranked list of 64,000 unique words in English was drawn from the British National Corpus.[23] As it turns out, the top 46 words are either unambiguous or are at the front of the candidate list (if there are collisions). They are entered without pressing next. The 47th ranked word is *if*, which is ambiguous with *he* at rank 15. So, while *he* is entered with keys 430, *if* requires 43n0. The top ten such words are shown in Figure 7.30. The issue is whether users adopt behaviors that negate or reduce the presence of mental operations. No doubt, expert uses will expeditiously enter top-ranked words without hesitating to monitor the display. This behavior may also extend to top-ranked words that require presses of next. While empirical evidence is lacking, anecdotal evidence suggests that experienced users press *next* without hesitation while entering words such as *if* and *no*.[24]

[23] ftp://ftp.itri.bton.ac.uk. The list is an expanded version of that used by Silfverberg et al. (2000). It is available on this book's web site.
[24] The author knows this from personal experience as well as from conversations with other mobile phone users on this subject.

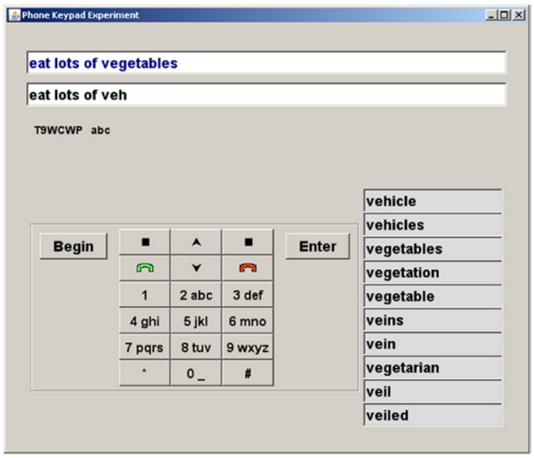


FIGURE 7.31

Input of *vegetables* using a GUI simulation of a phone keypad with word completion. After three key presses (834), *vegetables* appears as the third entry in the candidate list.

But how far down the list do such behaviors extend, and at what level of expertise do they begin? These are outstanding research questions. It is unlikely that any level of expertise would negate the need to monitor the display while entering *beep*, which is at rank 20,767 in the list. The word is simply too infrequent for sufficient skilled behavior to develop that would allow entry without mental operators.[25]

To move beyond keystroke-only modeling, heuristics are needed for the hidden costs—the mental operators. Two simple rules are *all-in* or *all-out*. An all-in model includes mental operators at every reasonable juncture (e.g., Figure 7.29). An all-out model recognizes an all-knowing absolute expert and excludes all mental operators. Assuming a reasonable value for the keystroking operator (K), the two approaches produce upper-bound (all-in) and lower-bound (all-out) execution time predictions. Neither is likely to occur in practice. More accurate predictions require a smarter heuristic, perhaps obtained through a creative pre-test.

Predictive text entry on a mobile phone is but one point in a rich and diverse problem space for text entry. Word completion adds to the mix by allowing entry of a full word after entering only part of it. A candidate list of complete words is generated with each key selection. If the desired word appears early, it is selected immediately, thus saving keystrokes. To explore mobile phone text entry involving word prediction and word completion, a Java GUI application was developed to test and demonstrate several setups.[26] The GUI simulation for word completion is shown in Figure 7.31.

[25]Of course, an obscure word that just happens to be entered frequently by a user would be expeditiously entered (i.e., no mental operators).
[26]The files, including an extensive API, and support files, are in `PhoneKeypadExperiment.zip`, available on this book's web site.

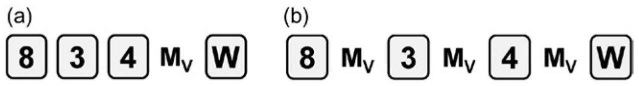


FIGURE 7.32

Key sequences and mental operators for input of *vegetables* (see Figure 7.31): (a) Visual check after third keystroke. (b) Visual check after each keystroke.

Keys are pressed by clicking on them with the mouse pointer or tapping on them with a stylus or finger. The entry of *vegetables* is shown. With each key press, a ten-word candidate list is generated.[27] After three key presses (834), *vegetables* appears in the third position in the candidate list. Clicking on it selects it, delivering it to the edit buffer. So four key actions produce 11 characters of text (*vegetables_*).

While there are fewer key actions, it is not certain that execution time or text entry speed will improve. What is a reasonable model for this interaction? Two possibilities are shown in Figure 7.32. If the candidate list is not generated until the third key action, or if the user chooses not to look until the third key action, the pattern in Figure 7.32a applies. The key action W is a selection on the candidate word. The mental operator M is coded as M_V, representing the time to visually scan the candidate words. It is clearly a different process than M_P—performing a physical match between a stimulus and a code in short-term memory. The visual search equation developed in Chapter 2 (section 2.7.2) is a logical substitute for M_V, but this is not explored here.

Figure 7.32b models a different scenario where the user is presented with and views the candidate list after each key action. Which behavior produces better performance? The behavior in Figure 7.32a is faster for the example (fewer operations), but what about English in general? This is a question for which an enhanced KLM can help. There are many design issues. A long candidate list bears a *cost* since it takes longer to visually scan but brings *benefit* since the desired word appears sooner. Viewing the list after each key action adds time to the interaction (*cost*) but allows entry of the intended word at the earliest opportunity (*benefit*).

The methods also differ between stylus or finger input and keyed input. For stylus or finger input, once the desired word appears and is located, it is selected directly. For keyed input, selecting the *n*th word in the candidate list takes an extra $n-1$ key actions, assuming arrows keys are required to navigate the list. And, as noted earlier, for mobile phone keyboards or stylus input, the pointing operator (P), in the form of a Fitts' law prediction, is more appropriate than the keystroking operator (K). Clearly there are many design choices and modeling issues that are ripe for a KLM analysis—one that includes a revised set of mental operators.

Models of predictive text entry in the HCI literature generally do not account for mental operations. The focus is usually on modeling or reducing keystrokes (M. Dunlop and Crossan, 2000; M. D. Dunlop, 2004; Gong and Tarasewich, 2005; Gong et al., 2008; MacKenzie, 2002; Pavlovych and Stuerzlinger, 2003; Ryu and Cruz, 2005).

[27] For convenience, discussions here speak of *key presses*. Since the interface uses soft keys, each key action involves a point-select operation using a mouse, finger, or stylus.

Models that predict text entry speed generally do so by incorporating only keystroking (K) or pointing (P) operators (MacKenzie and Soukoreff, 2002; Silfverberg et al., 2000).

7.2.4.4 Updating the KLM's mental operator (M)

Earlier I advocated updating the KLM's pointing operator (P) to include a Fitts' law prediction (P[A,W]). Perhaps the mental operator (M) also needs revisiting. Can the KLM's single mental operator (M) be replaced with a set of operators for the diverse interactions requiring user attention and cognition? M_P and M_V were suggested above as mental operations where the cognitive process involves physical matching or visual searching. But what are the execution times? What is the theory supporting these and other mental operators? Interestingly enough, the KLM's mental operator was not supported by any theory. The placement of KLM's mental operator involved heuristics to analyze the operations in a task and deduce where a mental operation was likely to occur (Card et al., 1983, p. 265). There was no theory offered on the cognitive processes or on the execution times expected for the operations, as is gleaned from experimental work in psychology or elsewhere. As noted, "The use of a single mental operator is a deliberate simplification" (Card et al., 1983, 263).

The value of M was estimated from the experimental data "by removing the predicted time for all physical operations from the observed execution time" (Card et al., 1983, 274). Then t_M was estimated by a least-squares fit of the time removed versus the estimated number of mental operations. The result was $t_M = 1.35$ s (*SD* = 1.1 s).[28] The standard deviation was quite large (81% of the mean), suggesting that the method of estimating M was rough, at best.

M_P, physical matching, was motivated by Card et al.'s model human processor (1983, ch. 2), which includes a range of human behaviors called "simple decisions." Notably, these behaviors are not included in the KLM. Each behavior involves a stimulus that is connected to a response through one or more cycles of cognitive processing. The behaviors are given in Figure 7.33, along with a proposed

Proposed Mnemonic	Task	Execution Time (ms)	
		Card et al.	Figure 2-28 & Figure 2-30
M_S	Simple Reaction	240 [105 – 470]	277 [±44]
M_P	Physical Matching	310 [130 – 640]	510 [±59]
M_N	Name Matching	380 [155 – 810]	485 [±52]
M_L	Class Matching	450 [180 – 980]	566 [±96]
M_C	Choice Reaction	$200 + 150 \log_2(N + 1)$	
M_V	Visual Search		$498 + 41\,N$

FIGURE 7.33

Mental operators and proposed mnemonics for simple decision tasks, along with the nominal value and the range of execution times (ms).

[28]The value $t_M = 1.35$ s was used to calculate the predicted execution times in validating the KLM. Given that t_M was estimated from observed data in the same experiment, the observed versus predicted comparison in Figure 7.21 is of questionable value, because the predictions are derived from the observations.

mnemonic. There are two sets of nominal execution times, one provided by Card et al. and another from the experiment described earlier in Chapter 2 (see Figures 2.28 and 2.30). The range appears in brackets as the upper-bound and lower-bound from Card et al.'s values or as ±1 standard deviation for the values in Chapter 2. The execution times include the time for motor response (e.g., a button or key press).

Card et al.'s values were obtained by reviewing a large body of literature on human sensory-motor responses. The range of execution times in brackets is large because of the variety of experimental procedures and task conditions included in their review. Some conditions increase execution time while others decrease execution time. For example, reaction times are known to vary by the intensity of the stimulus: Increase the loudness (auditory) or brightness (visual) of a stimulus and the reaction time decreases. The values from Chapter 2 are offered as an example. They are considered accurate but limited to the task, apparatus, and procedure used in the experiment. For M_S, M_P, M_N, and M_L, the values fall within the range provided by Card et al., although in all cases the value is greater than Card et al.'s nominal value.

For a simple reaction (M_S), the user is attending to the system, waiting for the onset of a stimulus. When the stimulus appears, the user reacts by pressing a key or button. For physical matching (M_P), the user compares the stimulus to a code stored in short-term memory. When the stimulus appears and a match is deduced between the stimulus and the code in memory, the user presses a key or button. An example was given earlier (see Figure 7.29). Name matching (M_N) is similar to physical matching except the user abstracts the stimulus in some manner, deducing equivalence. For example, the user might press a key or button in response to the stimulus of a "key" regardless of its appearance (e.g., "key" versus "**KEY**" versus 🔑). For class matching (M_L), the user makes multiple references from long-term memory. An example is responding to the stimulus of a letter regardless of which letter appears.

The nominal execution times for the top four mental operators in Figure 7.33 are in the range of 240–566 ms. All values are well below 1.35 seconds, the value for the KLM's original mental operator (M) (see Figure 7.20). At the very least, the addition of new operators—more aligned with a user's perceptual and cognitive processes—should help with KLM modeling of interfaces that include adaptive or predictive features. There is also considerable overlap in the execution times, in view of the lower and upper bounds. These are illustrated in Figure 7.34. So predictions of tasks or subtasks enhanced with these operators, while behaviorally correct, will vary considerably.

The visual search operator (M_V) cannot be expressed as a nominal value, or even a range of values. The time depends on the number of choices the user must scan. For the example in Figure 7.31, where the user is searching for a word among ten choices:

$$t_{M_V} = 498 + 41 \times 10 = 908 \text{ ms} \tag{24}$$

If this delay is unacceptable, it can be lowered by reducing the size of the candidate list to five words:

$$t_{M_V} = 498 + 41 \times 5 = 703 \text{ ms} \tag{25}$$

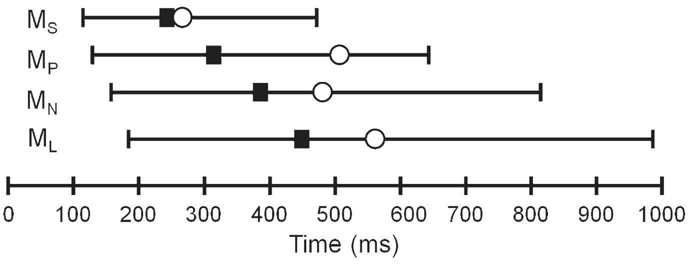


FIGURE 7.34

Nominal value and range of times (ms) for M_S, M_P, M_N, and M_L operators. Times are from Card et al. (1983, 65–76). Circles are means from experiment results in Figure 2.28 and summarized in Figure 7.33.

This time includes a match or no-match key press, which might be an over-simplification, depending on the interface. The difference of 205 ms is substantial and illustrates the kind of design trade-off that is possible if the range of interactions are analyzed and compared using a keystroke-level model. Let's consider another predictive model.

7.2.5 Skill acquisition

Not surprisingly, there is a relationship between skill and practice. Whether we are learning to play a musical instrument, fly an airplane, search a database, navigate a menu system, or touch type on a keyboard, we begin as *novices*. Initial performance is poor, but with practice we acquire skill. With continued practice, we become proficient, perhaps *experts*. If the skilled behavior is adequately represented by a ratio-scale variable, then the transition from novice to expert is well suited to predictive modeling.[29] In the model, the predicted or dependent variable is performance or skill, typically the time to do a specified task or the speed in doing the task. The independent variable is the amount of practice, typically in hours, days, years, trials, blocks, or sessions.

The relationship between skill and practice is non-linear. In the beginning, a small amount of practice yields substantial improvement. After a lot of practice, the same "small amount" produces only a slight improvement. It seems the best mathematical expression of the relationship is a power function of the form

$$y = b \times x^a \tag{26}$$

where x is the regressor or independent variable (amount of practice), y is the dependent variable (performance), and a and b are constants that determine the shape of the relationship. Represented in this form, the relationship between skill and practice is often called the *power law of learning* or the *power law of practice*

[29]This is generally not the case for playing a musical instrument or flying a plane. However, one can imagine unit tasks within these skills that are measurable with a ratio-scale variable and suitable for predictive modeling.

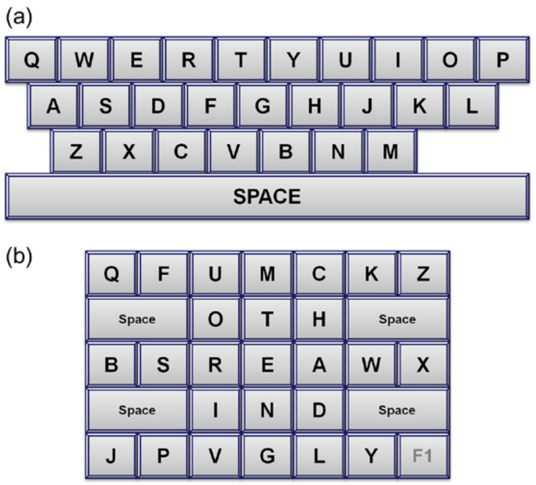


FIGURE 7.35

Soft keyboards layouts: (a) Qwerty. (b) Opti.

(Card et al., 1983, 27; Newell and Rosenbloom, 1981). The law dates at least as far back as the 1920s (Newell, 1990, p. 6).

If the dependent variable is the time to do a task, T, then Equation 26 can be recast as

$$T_n = T_1 \times n^a \tag{27}$$

where T_n is the time to do the task on the nth trial, T_1 is the time on the first trial. The trial number is n and a is a constant setting the shape of the curve. *Trial* is any practice indicator, such as task number, hours of practice, session number, etc. Note that a is negative, because the acquisition of skill means task completion time *decreases* with practice.

Alternatively, the dependent variable can be speed (S), the reciprocal of time. In this case, the model predicts "tasks per unit time," rather than "time per task." An example for speed might be text entry speed in "words per minute." Predicting speed, Equation 26 is recast as

$$S_n = S_1 \times n^a \tag{28}$$

where S_n is the speed on the nth trial, S_1 is the speed on the first trial, n is the trial number, and a is a constant setting the shape of the curve. In this form, a is positive since speed increases with practice. Furthermore, $0 < a < 1$ to reflect the diminishing return with practice. Let's develop an example.

An experiment compared two soft keyboard layouts for text entry using a stylus. (See Figure 7.35.) One layout had the familiar Qwerty letter arrangement. The other was Opti, a design intended to minimize stylus or finger movement for English text entry. The Opti layout was new to all participants. Because participants were familiar

(a)

Session	Qwerty	Opti
1	28.44	16.98
2	29.98	21.06
3	31.56	24.80
4	32.62	27.26
5	33.20	29.18
6	34.52	31.04
7	34.14	33.08
8	34.68	34.16
9	35.66	34.72
10	36.77	35.89
11	36.95	37.22
12	37.36	38.75
13	38.50	39.95
14	38.65	40.73
15	39.12	40.68
16	39.00	41.85
17	39.20	42.74
18	39.85	43.55
19	40.16	43.91
20	40.30	44.29

(b)

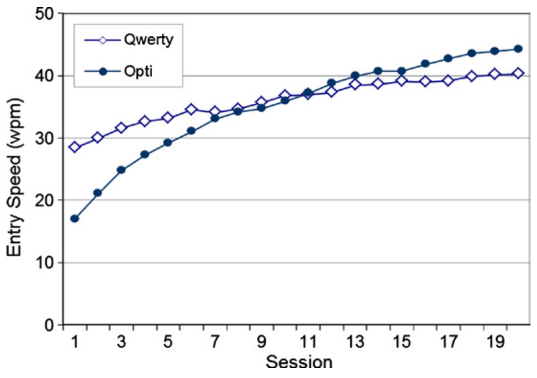


FIGURE 7.36

Experiment demonstrating skill acquisition: (a) Data showing entry speed (wpm) by session. (b) Chart.

with the Qwerty layout, better performance was expected for the Qwerty condition, at least initially. To test the potential of the Opti layout, a longitudinal study was conducted. With the acquisition of skill, would Opti eventually outperform Qwerty? If so, how much practice is required to reach a crossover point? The experiment involved 20 sessions of text entry. (See MacKenzie and Zhang, 1999 for complete details.)

Figure 7.36a gives the summary data from the experiment. The independent variables are practice, labeled "Session" in the table, and layout (Qwerty and Opti). The dependent variable is text entry speed in words per minute (wpm). Since two layouts were compared, there are two patterns of learning to consider. These are shown in Figure 7.36b. The improvement with practice in both cases follows the conjectured non-linear pattern. The crossover point—where Opti becomes faster than Qwerty—occurred at session 11.

The curve for the Qwerty layout is more flat because participants were well along the learning curve at the beginning of the experiment, due to familiarity with the letter arrangement. Curve fitting each set of points to Equation 26 is easily done with a spreadsheet application such as Microsoft Excel. The result is shown in Figure 7.37. Both curves show a power function with the observed points clustered close to the line.

The prediction equation for each layout is shown in the chart. The equations appear in their default power form (see Equation 26), as produced by Microsoft Excel. The goodness-of-fit is given by the squared correlation coefficient (R^2), commonly articulated as the percentage of the variance explained by the model. For Opti, 99.7 percent of the variance is explained by the model, and for Qwerty, 98.0 percent. These are very high figures and attest that both models are excellent predictors.

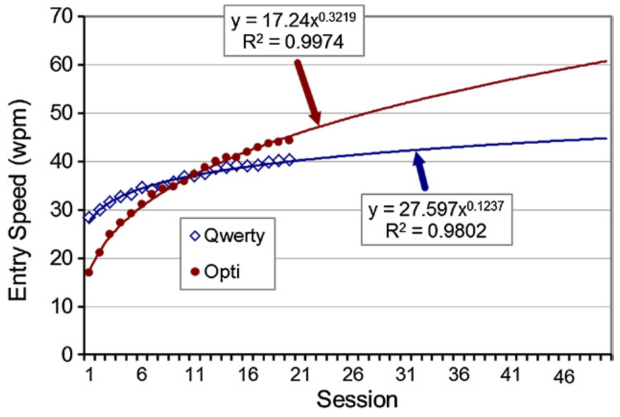


FIGURE 7.37

Power law of learning for Opti and Qwerty showing extrapolation to session 50.

Figure 7.37 also shows an extrapolation of the models to session 50. However, extrapolating beyond the testing range is risky. Although the correlations are high for both models, the ability to predict text entry speed diminishes the farther one ventures from the range of testing. For example, at session 50 the predicted text entry speed for Opti is:

$$S_{50} = 17.24 \times 50^{0.3219} = 60.7 \text{ wpm} \tag{29}$$

That's fast! Too fast, perhaps. Here's a simple exercise to demonstrate a model's ability (or inability!) to extrapolate. Using only the data for sessions 1–10 for Opti, the prediction equation is:

$$S_n = 17.03 \times n^{0.3319} \tag{30}$$

Using this equation to predict the text entry speed at session 20 yields:

$$S_{20} = 17.03 \times 20^{0.3319} = 46.03 \text{ wpm} \tag{31}$$

But the observed text entry speed at session 20 was 44.29 wpm (see Figure 7.36a). The prediction is generous by (46.03 – 44.29) / 44.28 = 4.0%. So perhaps the prediction for session 50 is generous as well.[30]

[30]Perhaps the prediction for Opti at session 50 for 60.7 wpm is realistic. In the experiment, text phrases were drawn at random from a small set of just 70 phrases (≈25 characters/phrase). The number of phrases per session was 50–60 early on, and 90–110 in later sessions. In view of this, participants were likely developing muscle memory with the phrases as the experiment progressed. This is similar to the highly over-learned motor skill one develops in other activities, such as buttoning a shirt, tying shoelaces, or entering a password (for people who never change their passwords). While this argument may have merit, it also means that the predicted text entry speed for session 50, while accurate, may apply only to the phrase set, not to English.

7.2.5.1 Skill acquisition research in HCI

Besides the experiment summarized above, there are other examples of the power law of learning in the HCI literature. The earliest is by Card, English, and Burr (1978), who compared a mouse, a joystick, and two keyboard techniques for selecting text on a CRT display. In building a learning model, they performed a log transformation on both the *x*-axis data (block) and the *y*-axis data (positioning time). With this change, the relationship is linear. The relationship for the mouse is shown in Figure 7.38. Similar log-log (linear) models are reported by Recker and Pitkow (1996) for mapping document access to document storage in a multimedia database, and by Seibel for a choice reaction task (reported in Newell 1990, 7).

Additionally, in *The Psychology of Human-Computer Interaction* (Card et al., 1983) there are skill acquisition models for reaction time (pp. 57–59) and execution time (pp. 284–285). There are numerous other sources in the HCI literature reporting predictive models of skill acquisition. Text entry is the predominant theme (Bellman and MacKenzie, 1998; Castellucci and MacKenzie, 2008; Clarkson, Clawson, Lyons, and Starner, 2005; Isokoski and Raisamo, 2004; Költringer, Van, and Grechenig, 2007; Lyons, Starner, and Gane, 2006; Lyons et al., 2004; MacKenzie et al., 2001; MacKenzie and Zhang, 1999; McQueen et al., 1995; Wigdor and Balakrishnan, 2003, 2004; Wobbrock, Myers, and Rothrock, 2006; Wobbrock, Rubinstein, Sawyer, and Duchowski, 2008; Zhai, Sue, and Accot, 2002). There are also examples in the literature where the progression of learning is presented in a plot but without including a model of the behavior (e.g., Itoh, Aoki, and Hansen, 2006).

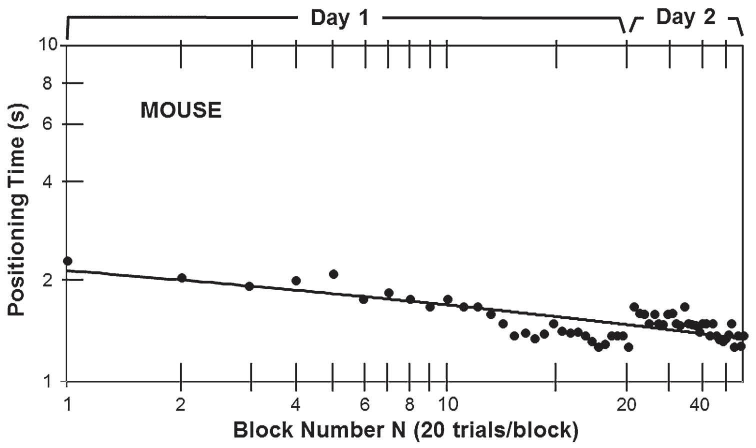


FIGURE 7.38

Skill acquisition for the mouse in a text selection task.

(Adapted from Card et al., 1978)

7.2.6 More than one predictor

The equations above predict the outcome on a dependent variable from a single predictor variable (independent variable). It is also possible to build a prediction model where the outcome is predicted from multiple predictors. In statistics the technique is called *multiple regression* and takes the form

$$y = a + b_1x_1 + b_2x_2 + b_3x_3 + \ldots \tag{32}$$

Let's begin with a hypothetical example. A researcher is interested in the relationship between the time taken to learn a computer game and the age and computing habits of players. Is age a factor in learning the game? Does the amount of daily computer use play a factor? Can the time to learn the game be predicted from a player's age and daily computer use? This last question suggests a prediction equation with multiple predictors. The dependent variable is y, the time to reach a performance criterion in playing the game. The two independent variables (predictors) are x_1, a player's age, and x_2, a player's daily computer use in hours. To explore the research questions above, an experiment was conducted. Fourteen participants were recruited. Their ages ranged from 16 to 40 years, and their use of computers ranged from two to eight hours per day. The participants were given basic instructions on the operation of the game. Then they were observed playing and learning the game in 15–20 minute segments over a period of one week. They continued to play until they reached a criterion score. The data are shown in Figure 7.39.

Participant	Time To Reach Criterion (hours)	Age (years)	Daily Computer Usage (hours)
P1	2.3	16	8
P2	2.1	17	7
P3	2.5	18	8
P4	3.6	23	6
P5	2.6	25	7
P6	5.3	26	5
P7	4.8	29	6
P8	6.1	31	4
P9	7.2	32	5
P10	7.3	35	3
P11	6.4	37	4
P12	8.1	38	2
P13	7.9	40	4
P14	2.3	16	8

FIGURE 7.39

Data for a hypothetical experiment testing the relationship between the time to learn a computer game and the age and daily computer use of players.

It is difficult to visually scan raw data and see patterns, so the first step is to import the data into a spreadsheet or statistics application and build scatter plots and other charts. Two scatter plots are shown in Figure 7.40. In both cases, a strong relationship is seen. Older participants generally took longer to reach the criterion in playing the game (Figure 7.40a). At $r = .9448$, the coefficient of correlation is high. The relationship is similarly strong between the time taken to reach the criterion score and players' daily use of computers. In this case, however, the relationship is negative. Players who self-report spending a lot of time each day using computers generally took less time learning the game (Figure 7.40b). The coefficient of correlation is $r = -.9292$, indicating a strong negative relationship.

Building a prediction equation using multiple regression is straightforward using any statistics application. Spreadsheet applications can also be used, depending on the features available. Microsoft Excel, for example, supports multiple regression using the Analysis ToolPak, which is available as an add-in. Regardless

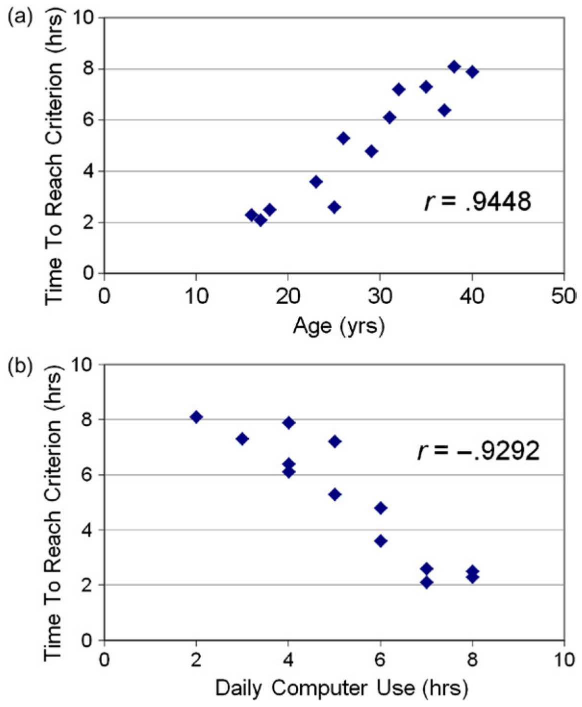


FIGURE 7.40

Relationships in the data between the time taken to reach a criterion score and (a) age and (b) daily computer use. The coefficient of correlation (r) is high in both cases.

of the tool used, a multiple prediction equation for the data in Figure 7.39 yields

$$y = 3.2739 + 0.1575x_1 - 0.4952x_2 \tag{33}$$

where y is the time taken to reach the criterion score, x_1 is a player's age, and x_2 is a player's daily computer usage in hours. The equation is accompanied with $R^2 = .9249$, meaning the model explains about 92.5 percent of the variance in the data. All in all, it's a very good model. Typically, other statistics are provided, such as the standard error of estimate for the prediction equation overall ($SE = .6769$, for the example above) and for each regression coefficient, and for the p-value for each coefficient.

As an example in the literature, MacKenzie and Ware (1993) studied the effect of lag (aka system delay or latency) on human performance while using a mouse in point-select tasks. The experiment followed a Fitts' law paradigm, manipulating target amplitude (A) and target width (W) in the usual manner. Lag was also manipulated by buffering mouse samples. Processing was delayed by multiples of the screen refresh period. The factors were target amplitude ($A = 96$, 192, and 384 pixels), target width ($W = 6$, 12, 24, and 48 pixels), and lag (8.3, 25, 75, and 225 ms). The A-W conditions yielded six levels of index of task difficulty (ID) ranging from 1.58 bits to 6.02 bits. The dependent variable was the movement time (MT in ms) to complete point-select tasks. The researchers presented four models for MT (see Figure 7.41). The first model in the figure is the traditional Fitts' law model. There is a single predictor, ID_e. The correlation is quite low, no doubt because of the additional variance introduced by the lag. The second model also uses a single predictor, lag. The fit is slightly better. The third model uses two predictors, ID_e and LAG. With $R = .948$, the fit is quite good.[31] The model explains 89.8 percent of the variance.

Model for *MT* (ms)[a]	Fit[b]	Variance Explained
$MT = 435 + 190\ ID_e$	$r = .560$	31.30%
$MT = 894 + 46\ LAG$	$r = .630$	39.80%
$MT = -42 + 246\ ID_e + 3.4\ LAG$	$R = .948$	89.80%
$MT = 230 + (169 + 1.03\ LAG)\ ID_e$	$R = .987$	93.50%
[a] *LAG* in ms, *IDe* in bits		
[b] $n = 48$, $p < .0001$ for all models		

FIGURE 7.41

Models for movement time (MT) based on index of difficulty (ID_e) and system lag (LAG).

(From MacKenzie and Ware, 1993)

[31] It is a convention for multiple regression to use uppercase R for the correlation coefficient.

The final model in Figure 7.41 reorganizes the independent variables to demonstrate an observed interaction effect between lag and task difficulty. Indeed, there is an improvement in the model: $R = .987$, explaining 93.5 percent of the variance. At lag = 0, the fourth model reduces to

$$MT = 230 + 169 \times ID_e \tag{34}$$

which is consistent with other Fitts' law models for the mouse. Each millisecond of lag in the fourth model adds 1 ms/bit to the slope of the prediction line. This is the sort of additional explanation that motivates adding extra predictors to regression models.

Similar uses of multiple regression in the HCI literature are reported by Accot and Zhai (2001), Cao et al. (2008), Rohs and Oulasvirta (2008), and Sears and Shneiderman (1991).

HCI researchers with a social science perspective tend to use multiple regression with a slightly different motivation. Social scientists are more interested in observing and explaining human behavior rather than in measuring and predicting human performance. Typically the research seeks to determine the relative impact of several behavioral factors (independent variables) on a dependent variable. A common method is *stepwise linear regression*. All variables are tested individually to determine which has the greatest explanatory power (highest R^2) on the variance in the dependent variable. A linear model is built using that variable. Then the remaining variables are tested one at a time against the model to determine which has the greatest explanatory power on the remaining variance. The process is repeated until all variables are added. For example, Dabbish et al. (2005) describe a model using "probability of replying to an e-mail message" as a dependent variable. The model was built using 12 predictor variables. Some contributed significantly and positively to the dependent variable (e.g., number of e-mails with only one recipient), some contributed significantly and negatively to the dependent variable (e.g., number of e-mails from close colleagues), while others had little or no effect (e.g., number of e-mails about scheduling). Even with 12 variables the model had a modest $R^2 = .37$. However, behavioral variables are, in general, less stable than performance variables, so overall Dabbish et al.'s model was good. Other researchers describe similar stepwise multiple regression models (Chung and Hossain, 2008; Iqbal and Bailey, 2006; Lampe, Ellison, and Steinfield, 2007; Seay and Kraut, 2007; Shklovski et al., 2006; Su and Mark, 2008).

One property of multiple regression deserves mention. When an additional predictor is added to the model, the correlation (R) always increases. It cannot decrease. Thus, caution is warranted in declaring the new model a "better model" simply because of a higher correlation.[32] The true test is whether the new predictor offers additional explanatory power relevant to the intended application of the model. MacKenzie and Ware's work was motivated to model interaction in virtual reality systems, where lag is common; thus, adding lag to the model makes sense.

[32]For example, if participants' shoe sizes were measured, then shoe size (SS) could be added as a predictor. The model's correlation would increase!

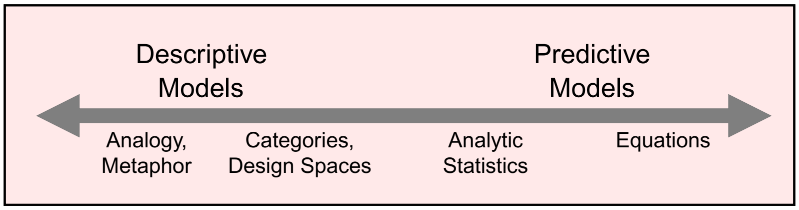


FIGURE 7.42

Model Continuum Model (MCM) with descriptive models at one end and predictive models at the other end.

7.3 A model continuum model

In this chapter, I have treated the problem space of modeling as a dichotomy—a space with two mutually exclusive zones: descriptive models and predictive models. While useful, this is in conflict with the model continuum suggested by Pew and Baron, which we examined at the beginning of the chapter. If the space is a continuum rather than a dichotomy, there will be identifiable points along the way. I will wrap up this chapter by proposing another model, a Model Continuum Model (MCM). (See Figure 7.42.)

Figure 7.42 is at best a suggestion, a work in progress. Certainly it lacks the organization and clarity of the descriptive model of politics in Figure 7.1. The model shows a continuum and places descriptive and predictive models at opposite ends. It suggests positions along the continuum where certain kinds of activities or processes might be placed according to their descriptive or predictive emphasis. Are there ways to improve the model? Perhaps the terms *qualitative* and *quantitative* should be added at each end.

This chapter presented the core ideas of descriptive and predictive models of human-computer interaction. The student exercises that follow take up and extend many of the ideas presented above.

STUDENT EXERCISES

7-1. Below is a "big fuzzy cloud" model of human-computer interaction.

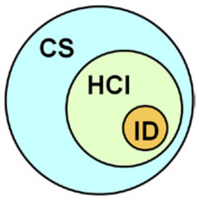
Improve on this by proposing a descriptive model of human-computer interaction as per the discussions in Section 7.1.1: Delineating a Problem Space (see Figure 7.1).

7-2. Propose a descriptive model of text entry. (Hint: Do not try to encompass all aspects of text entry. Work with only two or three aspects of text entry that suggest a delineation of the problem space.)

7-3. Propose a descriptive model of ______________. (Fill in with any HCI research topic.)

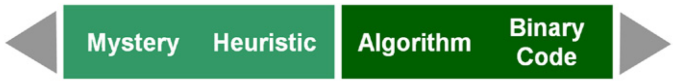
7-4. Review published papers in HCI to find a descriptive model. Summarize the model. Identify at least one weakness or limitation in the model and propose at least one improvement or extension to the model.

7-5. At a recent conference on human-computer interaction, a presenter used a Venn diagram to demonstrate the relationship between the disciplines of computer science (CS), human-computer interaction (HCI), and interaction design (ID):



Critique this descriptive model. Suggest a revision to the model. Perhaps you can add one more circles to the Venn diagram to include an additional discipline or activity.

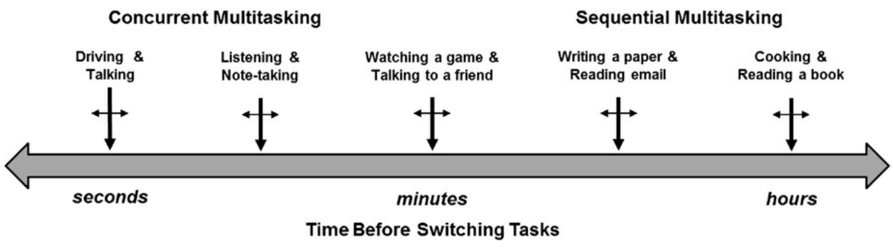
7-6. The following is a descriptive model for the design process (from Thornton, 2010). The model shows a continuum of activities that encompass design according to the philosophy of *design thinking* (Brown, 2009; R. L. Martin, 2009).



Design Thinking **Continuum**

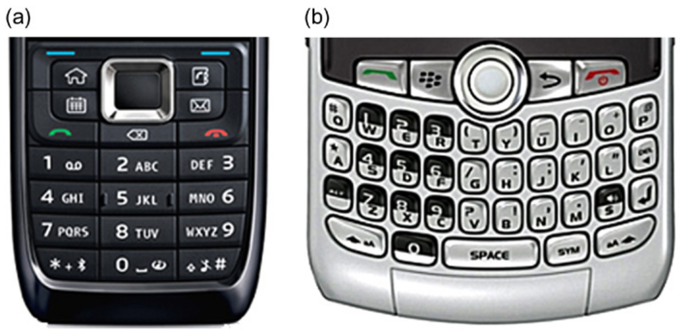
Write a brief analysis of this model considering the following: What is *design thinking*? What is meant by *mystery*, *heuristic*, *algorithm*, and *binary code* as applied to design thinking? Why do the labels appear in two groups? Are there activities beyond each end of the continuum, as inferred by the arrows? (Hint: Search Google Scholar using the phrase "design thinking.")

7-7. The following is a descriptive model of multitasking (adapted from Salvucci, Taatgen, and Borst, 2009). The model shows a series of task pairings along a time continuum with concurrent multitasking at one end and sequential multitasking at the other end.



Write a brief analysis of this model considering the following: What is *multitasking*? What are *concurrent multitasking* and *sequential multitasking*? Identify at least three additional task pairings, where at least one of the tasks involves humans interacting with technology, and place each one in the model. Also, identify a task pairing that is positioned beyond the left extreme of the model and another beyond the right extreme of the model.

7-8. Besides the desktop keyboard, the key-action model (KAM) applies to keyboards on mobile devices:



Create an illustration similar to Figure 7.3 for a mobile phone keyboard, such as above. Preferably use your own device. Propose some changes to the model to make it more representative of the keyboard.

7-9. Beaudouin-Lafon defines an *interaction model* (briefly) as "a set of principles, rules, and properties, that guide the design of an interface" (Beaudouin-Lafon, 2000, p. 446). Where are interaction models positioned in the "model continuum model" shown in Figure 7.42? Write a brief review of both models and propose a revised pictorial of the model continuum model that includes interaction models and other models reviewed by Beaudouin-Lafon.

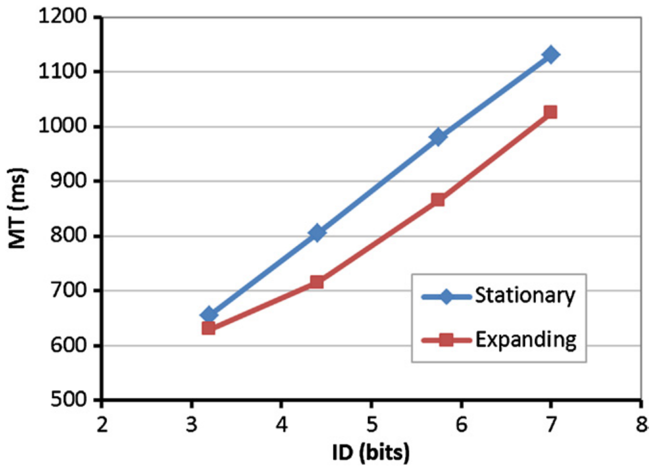
7-10. Jacob et al. (2008) propose a framework for a category of interaction styles called Reality-Based Interaction (RBI). Their framework is a descriptive model. Summarize RBI and demonstrate how it meets the criteria described above for a descriptive model. Give examples.

7-11. The following data show the stylus tapping speed (S_{ST}) and touch typing speed (S_{TT}) for ten participants. Both variables are in words per minute (wpm).

Participant	Stylus Tapping Speed (wpm)	Touch Typing Speed (wpm)
P1	21.4	42
P2	23.6	44
P3	22.0	32
P4	24.0	50
P5	23.0	36
P6	17.1	33
P7	29.0	55
P8	14.7	22
P9	20.3	31
P10	19.7	33

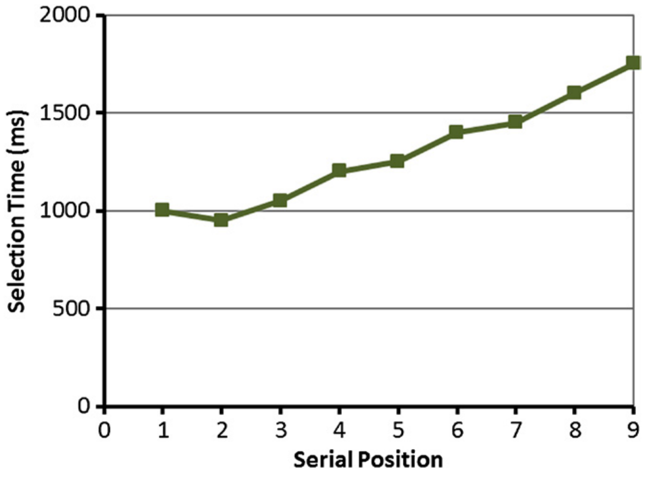
What is the coefficient of correlation (r) between the two variables? What is the linear prediction equation for stylus tapping speed, given touch tapping speed? What percentage of the variation (R^2) is explained by the prediction model? Create a chart showing a scatter plot of the points, the prediction equation, and R^2.

7-12. The following chart was adapted from a Fitts' law study comparing target selection times for expanding and stationary targets (adapted from Zhai, Conversy, Beaudouin-Lafon, and Guiard, 2003). (Note: Expanding targets increase in virtual size when the cursor is close, thus potentially improving interaction.)



The authors did not build predictive models for the data sets, but they could have. Use a ruler or some other apparatus (e.g., Microsoft PowerPoint or Adobe PhotoShop) to reverse engineer the chart and create data sets for the Expanding and Stationary conditions. Enter the data into a spreadsheet application, such as Microsoft Excel. Create a new chart that, for each condition, includes the regression line, prediction equation, and R^2. Which condition is better in terms of *Throughput*? Explain.

7-13. The following chart was adapted from a study measuring the selection time for menu items based on the serial position of an item within the menu (adapted from Byrne, Anderson, Douglass, and Matessa, 1999, Figure 1).



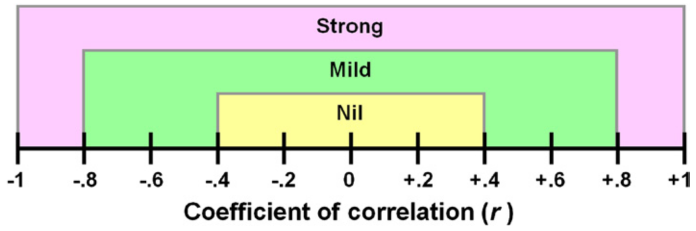
The scatter plot appeared without a regression line. Reverse engineer the chart and build a data set for the scatter plot. Recreate the chart, showing a regression line, regression equation, and R^2. What is the predicted time to select the fifth item in the menu? What is the 95 percent confidence interval for the prediction?

7-14. The data from a Fitts' law study with a mouse are given below:

A (pixels)	*W* (pixels)	*ID* (bits)	*MT* (ms)
192	16	3.70	654
192	32	2.81	518
192	64	2.00	399
320	16	4.39	765
320	32	3.46	613
320	64	2.58	481
512	16	5.04	872
512	32	4.09	711
512	64	3.17	567

Using a spreadsheet application, create a chart showing the scatter plot, regression line, prediction equation, and R^2. (Note: These data were used for Figure 4a in Wobbrock et al., 2008.)

7-15. Users and computers (Part III). Extend the report for Part II of this exercise (see Chapter 6) to determine if there is a relationship between the age of respondents and the number of hours per day of computer use. Since both variables are ratio-scale, the relationship may be explored using a scatter plot, regression line, and the coefficient of correlation (*r*). For the purpose of the report, consider the relationship strong, mild, or nil according to the following criteria:



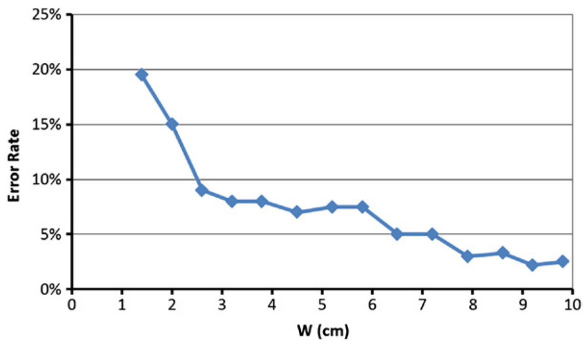
If the relationship is strong, propose a prediction equation that gives the number of hours per day of computer use as a function of a user's age.

7-16. The data from a choice reaction time experiment are given below:

Choices (n)	Reaction Time (ms)
1	155
2	265
3	310
4	365
5	400
6	425
8	505
10	530

Plot the data and build regression models using a few different relationships (e.g., linear, power). What model explains the most variation in the data? (Note: These data were reverse engineered from Figure 4 in Seow, 2005.)

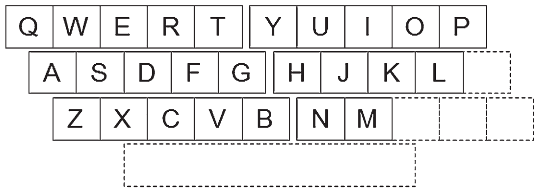
7-17. The following chart was adapted from an experiment showing the relationship between error rate and target width in a target selection task using a camera phone (Rohs and Oulasvirta, 2008):

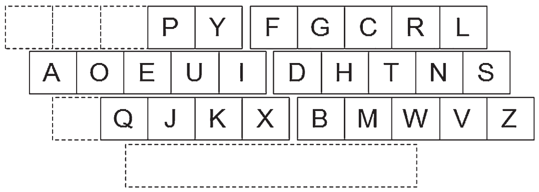


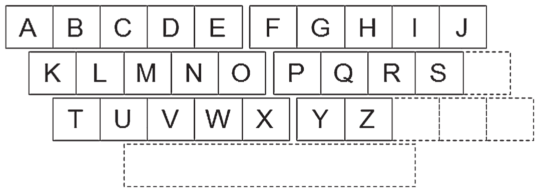
As expected, smaller targets tend to be incorrectly selected more often than larger targets. The relationship seems well suited to curve fitting using a power formula. Reverse engineer the chart and generate a set of *x*-*y* points where $x =$ target width and $y =$ error rate. Use a spreadsheet application to recreate the chart showing the scatter of points and the best-fitting curve. What percent of the variation in the observations is explained by the model?

7-18. A human operator attends to eight stimulus lights and presses one of eight keys when the corresponding light turns on. Two of the lights turn on more frequently than the others, accounting for 40 percent and 30 percent of all activations, respectively. The other lights activate with the same frequency. What is the information content of the task?

7-19. Below are the layouts for the Qwerty and Dvorak keyboards, as well as an alphabetic layout proposed by Card et al. (1983, 63). Assuming the layouts are implemented as standard physical keyboards, which design provides the most even split between left-hand and right-hand keying? (See Figure 7.19.) Propose a new design where the split is more even between hands.



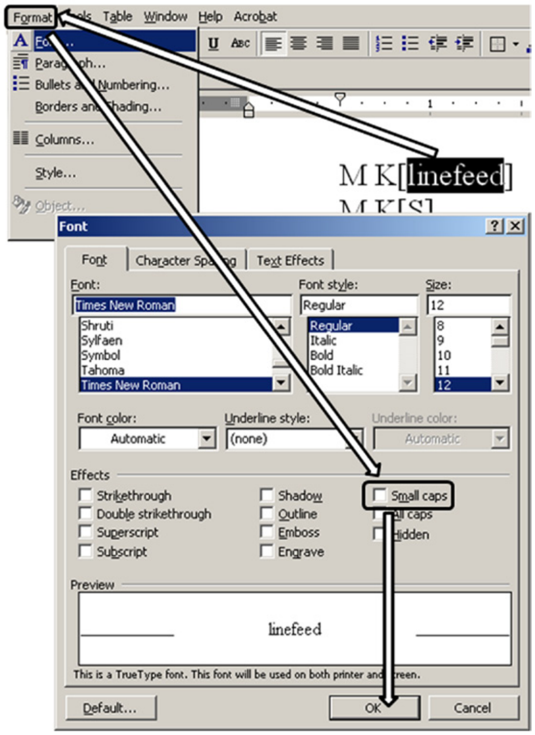




7-20. If the keyboards above are implemented as software keyboards on a desktop computer, the keys can be accessed (pressed) using a mouse. Use the Fitts' law mouse model given earlier (Equation 12) to compute the time to enter *the quick brown fox jumps over the lazy dog*. Convert the time to an entry speed in *words per minute* (wpm). Repeat for the Dvorak and Alphabetic layouts. Repeat using the RemotePoint Fitts' law model (Equation 11). In digitizing the keyboards, ignore the small gaps between keys in the images.

7-21. Repeat the exercise above for the Opti soft keyboard layout in Figure 7.35b. Of the four SPACE keys on Opti, assume the user chooses the one that minimizes the movement distances (e.g., for C_W, the user chooses the SPACE key at the top right).

7-22. Below is an illustration of a sequence of mouse operations to set "Small caps" as the formatting style for a word of text. The screen snap is from MS Word:



Font: Times New Roman
Font style: Regular
Size: 12
Font color: Automatic
Underline style: (none)
Underline color: Automatic

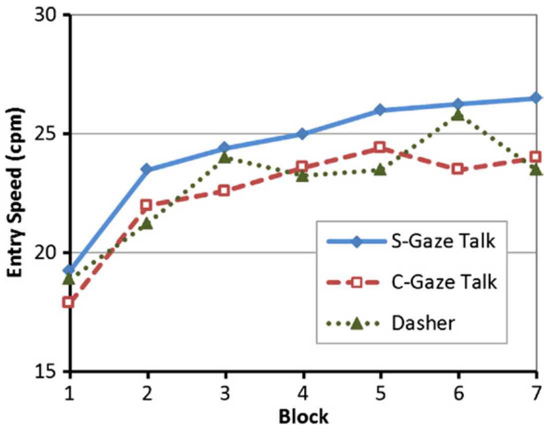
This is a TrueType font. This font will be used on both printer and screen.

What is the predicted execution time for the task? What is the predicted execution time if the keyboard is used instead of the mouse? For this question, assume $t_K = 0.4$ seconds. For both questions, provide a KLM breakdown of all operations.

7-23. Use the KLM to predict the time to enter "I hate baking pies" on a mobile phone. Provide three predictions: one for multi-tap, one for predictive input using an "all-in" assumption for mental operators, and one using an "all-out" assumption for mental operations. In building the models, assume the keystroking operator (K) is nominally $t_K = 0.4$ seconds. For predictive input, the ordered collision sets at the ends of the words are as follows: I = {I}, hate = {have, gave, gate, hate}, baking = {baking, baling, caking}, and pies = {pier, pies, rids}.

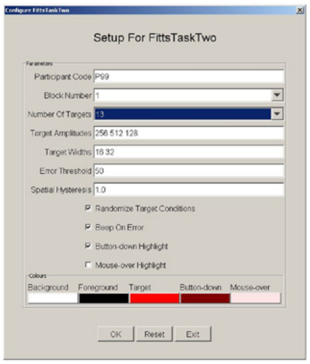
7-24. Build a multiple regression model for the data in exercise 7-14 above. Treat *MT* as the dependent variable and *A* and *W* as predictors. What is the multiple regression equation? What is the percent of the variance explained by the model? Experiment with some transformations on A and W to obtain a better model (higher R^2). What transformation produces the highest R^2?

7-25. The following chart shows the relationship between text entry speed in characters per minute (cpm) over seven blocks of testing for three entry methods: GazeTalk (standard), GazeTalk (centered), and Dasher (adapted from Itoh et al., 2006, Fig. 4). The power law of practice is clearly evident in the chart; however, no model was provided in the publication.



Reverse engineer the chart and generate a set of x–y points for each of the three entry methods. Use a spreadsheet application to recreate the chart showing the scatter of points and the best-fitting curve and prediction equation for each method. What percent of the variation in the observations does each model explain?

7-26. Conduct an experiment comparing two input methods for pointing and selecting. Use the `FittsTaskTwo` software from this book's website. Use any two methods of input. Consider using two input devices or a single input device operated in different ways. The setup dialog and a screen snap of the experiment procedure are shown below.


Configure FittsTaskTwo

Setup For FittsTaskTwo

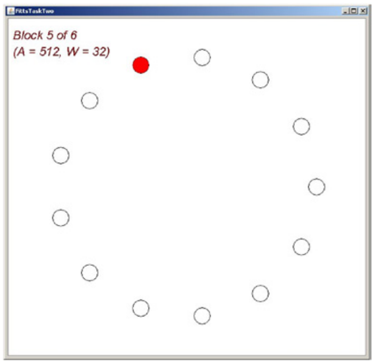
Parameters

Participant Code	P99
Block Number	1
Number Of Targets	13
Target Amplitudes	256 512 128
Target Widths	16 32
Error Threshold	50
Spatial Hysteresis	1.0

[✓] Randomize Target Conditions
[✓] Beep On Error
[✓] Button-down Highlight
[ ] Mouse-over Highlight

Colours

Background	Foreground	Target	Button-down	Mouse-over
⬜	⬛	🟥	🟥	⬜



The software implements the multi-directional ISO 9241-9 protocol for evaluating non-keyboard input devices. Consult the API for complete details. Configure the software using three levels each of target amplitude and target width, with 13 targets per sequence (see above). Ask the participants to perform five blocks of trials for each condition. Consider other aspects of the experiment design as per discussions in Chapter 5 (Designing HCI Experiments). Write a brief report on your findings. Among the results to report, include ANOVAs for the effects of input method and block on movement time, throughput, and other dependent variables. Also include a chart with scatter data, regression lines, prediction equations, and squared correlations, as per Figure 7.17.