Chapter 7:

THEORIES, MODELS AND BASIC CONCEPTS

Introduction

In this chapter we cover some of the basic theories and models that form the underpinnings on which the rest of the book are based and in the next chapter, we go into some issues of human performance in more detail. For many readers, some of the concepts will be "old hat," somewhat akin to finding a discussion of arithmetic in a book devoted to mathematics. For those readers, we suggest skipping the section in question. The nature of the field is interdisciplinary, and one reader's old news is another's new and novel concept. The key thing here is that after these two chapters, there will be a basis of common vocabulary and literacy that we can assume, and which can serve as the basis for further discussion.

In what follows, we move more-or-less progressively from the motor-sensory to the cognitive systems. However, if there is one editorial position that we want to emphasize throughout, it is that *you should not discuss one without the other.* That is, the literature to date as been split between that which applies to the motor-sensory level (often referred to the *pragmatic* or *device* level) and the *cognitive* level (learning, cognitive models, etc.).

What we argue is that the pragmatic level plays a crucial role in shaping the cognitive model of the system, and that in designing the pragmatic level, you are in effect shaping the mental model. Stated another way, if one has a clear idea as to the mental model one wants to conjure up in the user, then the affordances of the pragmatic level constitute the most effective way of doing so.

Time-Motion Studies: From Taylor to the Keystroke-Level Model, Fitts' Law and the Steering Law.

Introduction

Towards the end of the century, a number of researchers from a range of backgrounds became interested in human motion and efficiency of action. This interest stemmed from both scientific curiosity and interest is specific applications – applications ranging from improving human performance in athletics to make workers more efficient at their job. The tools and methodologies used in such studies, and included photography, the stopwatch, and measurement instruments

that very much resemble the motion capture devices used in today's animation and video-game studios.

The American Eadweard James Muybridge (1830-1904) and French physiologist Etienne-Jules Marey (1830-1904) were prime early figures in the use of photographic techniques[1]. Muybridge, the better known of the two, was more of an artist than scientist. Marey's approach was more scientific, even if his purpose was largely applied, and he made significant contributions to the development to technologies and techniques to support the study of human and animal motion – from the beat of the heart to how we (and horses) walked and ran.

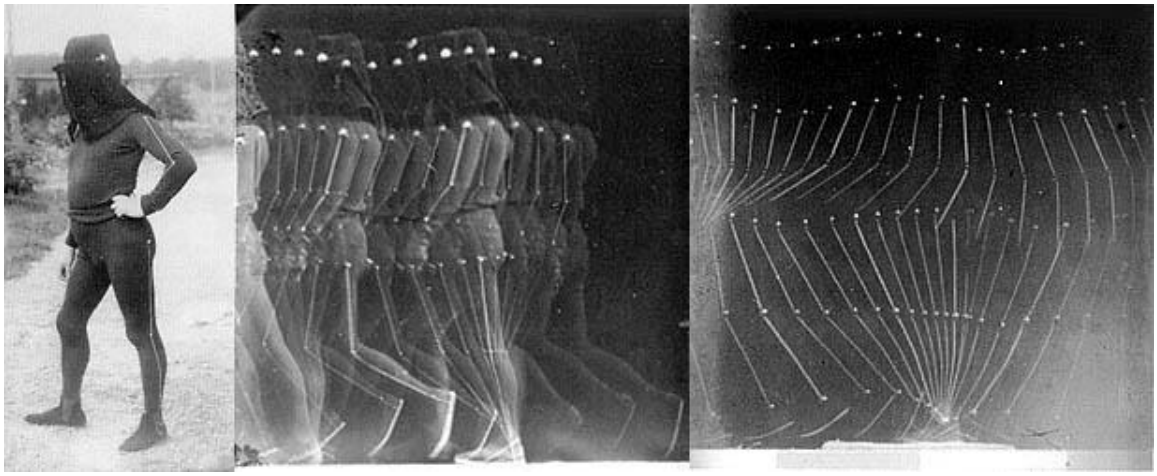

Figure 1: Studying Human Motion with Time-Lapse Photography.
To understand the motion of walking humans, Marey dressed people in black costumes and photographed them from a single location, with a single camera, doing exposures at regular time intervals. The black suits had light lines along the sides of the outside limbs, so what one ended up with is a visualization that consists of just the movement of the limbs of interest represented over time. He used similar techniques to capture the motion of other activities, such as tracking the motion of the hands and arms in manual work. (Photo from Braun, 1992).

Meanwhile, the American Frederick Winslow Taylor (1856-1915) was laying the foundation for what he called *Scientific Management,* which was an approach to organizing work that put efficiency of process above all. Taylor's work was picked up and refined by Frank Gilbreth (1868-1924) and his wife, the psychologist Lillian Moller (1878-1972).[2]

Of interest to us in this work is the application of scientific rigor to the understanding of motion in time. Besides the application of scientific techniques, what is important in this work is that it was applied to describing what was happening (the analytic component) as well as to derive improved methods for performance (the proscriptive component). Hence, in one case the analysis of performance might be used to improve the process of laying bricks, while in another it might lead to understanding how an athlete can jump higher or farther.

---

[1] As an aside, given the overlap in interests of these two men (such as using photography to study horse movement), it is interesting to note that their births and deaths were in the same year, and they shared the same initials.

[2] The interested reader is directed to Braun (1992) for an outstanding study of the work of Marey, as well as an excellent overview of the work of Muybridge, Taylor and the Gilbreths, as they apply to the organization of work. An excellent introduction to the work of Taylor can be found in Kanigel (1997), while those interested in reading the original are directed to Taylor (1947). Those interested in the work of the Gilbreths are directed to a collected volume of their writings edited by Spriegel and Myers (1953).

Not surprisingly, this general approach has been applied (adopted may be a better description) to the study of human-computer interaction. As in the early work outlined above, the techniques can, and have been, adapted integrated into the process of informing the design of today's digital technologies, and how we interact with them. They can be applied to figure out what is wrong with a design as well as provide the basis for developing new techniques. Used appropriately, they can also provide a useful tool for comparing alternative designs – be they existing techniques which can be measured, or by simulation of candidate designs, using empirically derived models, rather than actual implementation.

This last point is important. Without any question, user testing and iterative design, supported by sketching and prototyping are a critical part of the design / product development process (Buxton & Sniderman, 1980; Tohidi, Buxton, Baecker & Sellen, 2006; Buxton, 2006). However, there are two important points to keep in mind before overemphasizing an accompanying "learn by making" attitude.

First, iterative "making" initiatives can benefit significantly by the incorporation of evidence-based models during the planning and evaluation phases. Often, quick, simple, inexpensive back-of-the-envelope calculations can help filter design options to better identify the best candidates for prototyping.

Second, "making" (or "synthesis") should be balanced by "sampling" in the process of enumerating candidate design approaches. Furthermore, empirically derived models, such as those discussed in this chapter, can play an effective role in evaluating this class of this too-neglected pool of historical[1] candidates.

In short, the application of theory, which evolves out of the science of design, is as important to effective design, as all of the other more conventional techniques which we consider when we think about design. Both are essential. Neither is sufficient.

The Keystroke-Level Model

Introduction

In the area of time-motion study as applied to input to computer systems, perhaps the most important and useful tool is the *Keystroke level model* of Card, Moran and Newell (1980).

The Keystroke-Level Model was developed by Card, Moran and Newell (1980), and grew out of research at Xerox's Palo Alto Research Center (PARC). It provides a means for predicting one aspect of human performance when interacting with computers: how long it will take an expert user to perform a particular task on a particular system, assuming error free performance.

It is equally important to state what the model does *not* predict: errors, learning time, retention, system functionality, fatigue, acceptability, or the concentration required. While important considerations, these are not issues addressed by this particular model.

To use the model to predict how long an expert would take to perform a particular task on a particular system, one breaks it into its constituent elements, what Card, Moran and Newell call *unit tasks.* What is actually broken down in order to derive this recipe of unit tasks is the *method* that would be employed if an expert user was to perform the given task on the given system. Simply stated, the predicted time to perform the overall task is the sum of the times to perform each of the constituent unit tasks, following this method.

Assume, for example, that the task to be performed is to create a web page, and the designer wants to do a paper-and-pencil comparative evaluation of two different interface designs. Using

[1] It is difficult to overemphasize the importance of cultivating a strong knowledge of, appreciation for, the history of interactive techniques. One should enter the design process with the belief that anything that you will think of has been done before, somewhere, in some meaningful way (Buxton, 2008).

the model, the designer could analyse the method that an expert would follow, using each interface under consideration. Each method would then be broken down into its constituent unit tasks, and a time for each predicted. Summing the times determines the total task time for each method.

While the basic concept seems simple, as usual, the devil is in the detail.

Unit Tasks

Unit tasks, according to the model, are at the level of pushing a key on the keyboard, hence the name. Each unit task, *T,* is made up of two parts: (1) *acquisition* and (2) *execution,* and their relationship can be more formally expressed as:

$$T_{task} = T_{acquire} + T_{execute}$$

In colloquial terms, these might correspond to figuring out what to do, and doing it. Thus, the time spent looking at a document and determining what you want to change would constitute task acquisition, whereas actually making the changes would be the execution phase.

Task acquisition time very much depends on the nature of the work being done. When typing a handwritten manuscript, acquisition would only take the second or two that it takes to read the next chunk of text to be typed. Once typed, in revising that same document, acquisition time might well be in the 10s of seconds, since one might well want to ponder what the best wording of the change might be. And finally, if the document was a novel that the author was composing at the keyboard, the acquisition time might be an hour while the fledgling author thought up the perfect line of dialogue for the protagonist.

This is all by way of saying that while task acquisition is important and effects overall task performance, it is not something that we can predict and control for, and therefore, it is not taken into account by the keystroke-level model.

Hence, in what follows, we will be discussing the execution phase of unit tasks only.

Operators

The execution phase of a unit task can be categorized in terms of one of six basic *operators.* The first four are physical-motor based, the fifth cognitive, and the sixth has to do with the system response. They are:

K: (Keystroking). This refers to all keystrokes, regardless of where the key is located or what its function is. It could be a key on the QWERTY keyboard, a function key or a mouse button. Since it includes button pushes as well as *SHIFT*, *SPACE*, etc., the number of keystrokes is not the same as number of characters typed.

P: (Pointing). This is the task of pointing at a target on the screen. This may be accomplished using a number of different devices (such as a mouse, joystick, trackball, etc.) using a number of different techniques. The discussion of Fitts' Law later in this chapter lays the foundation for making appropriate estimates for this operator.

H: (Homing). This represents that time required to get into "home position" for the device on which the next operation will be executed. Imagine that the previous task was a **K** operation and the next one involves a pointing (**P**) operation. The **H** operator takes into account the time taken to move from the keyboard to the pointing device.[1]

[1] This is an important issue. As we shall see in the discussion on Fitts' Law, later in this chapter, there are many studies that compare the pointing times of different devices. But few, if any, take into account the "overhead" of acquiring the device (what the **H** operator accounts for). For example, from the literature one

D: (Drawing). At the time that the Keystroke-Level Model was first published (1980), interactive graphics applications were not too common. Hence, the notion of drawing described in the paper is somewhat limited. It assumes drawing what are now called *rubber-band lines.* However, subsequent work (MacKenzie, Sellen & Buxton, 1991) has shown that other tasks – most notably *dragging* tasks - are in this same class as Card, Moran and Newell's drawing task, and that these, too, can be modeled using Fitts' law, as is discussed later in the chapter. Most conveniently, dragging also starts with a "D", so the folding this general class of tasks under the **D** operator is less confusing than it might have been.

M: (Mental). This operator is a placeholder whose purpose is to account for the time taken to mentally prepare for certain tasks. While it is true that expert performance is characterized by *automatic* execution, there is still planning and decision making that consumes time and cognitive load between such automatically executed chunks.

R: (Response). Computers are getting faster and faster. But then, we are also asking them to do more and more. Consequently, response is not always instantaneous. Hence, the **R** operator is included in order to account for that portion of execution time that is a consequence of waiting for system response.

Allen, R.B. & Scerbo, M.W. (1983). Details of command-language keystrokes. *ACM Transactions on Office Information Systems,* 1(2), 159-178.

Fitts' Law

Introduction

Fitts (1954) ran some experiments to study the effect of target size and distance on target acquisition time. (That is, how long did it take to point at something?) Of interest to us is an experiment involving a one-dimensional reciprocal tapping task. Subjects were presented with two bars, as illustrated in Figure 2 and tapped back and forth between the two as quickly as possible.

The results of these experiments, known as *Fitts' Law*, state that the time to acquire a target with a continuous linear controller has a logarithmic relationship to the distance over the target size. Stated more formally, the movement time *MT* to move the hand to a target of width *W* which lies distance (or amplitude) *A* is:

$$MT = a + b \; I\,o\,g_2(\,2A/\,W) \qquad (1)$$

where *a* is a constant, and (according to Card, Moran & Newell,1983, p. 241) *b* = 100[70~120] msec/bit.

A key concept of Fitts' formulation is that of the *index of difficulty* (ID) that is expressed as:

$$ID = I\,o\,g_2(\,2A/\,W) \qquad (2)$$

That there is something wrong with Fitts' formula emerges when one starts to look at the values for *ID* reported in a number experiments. In order to match Fitts' Law with the data, many such studies report an I*D* that is negative! (A number of examples of this can be found, for example, in the proceedings of various SIGCHI conferences.) If nothing else, the notion of "negative difficulty" is inelegant; so how has this come about?

---

might well conclude that a stylus is a faster pointing device than a mouse for a particular application. But the homing time is significantly higher with a stylus than a mouse, so if most **P** operations are preceded by a **K** operation, it may turn out that the efficiency of pointing with the stylus are more than counterbalanced by the higher overhead in **H**. Herein lies one of the key benefits of the Keystroke-Level model. It is low level enough to be tractable, yet deals with things at a high enough level to capture such contextual issues.

Fitts' Law was based on work in information theory by Claude Shannon (Shannon and Weaver, 1949). MacKenzie (1989, 1991 and 1992) argues that Fitts made an unnecessary deviation from Shannon's work and proposes the following reformulation of the law, more accurately reflecting Shannon's work:

$$MT = a + b \log_2 (A/W+1) \qquad (3)$$

or, expressed in terms of *ID:*

$$ID = \log_2 (A/W + 1) \qquad (4)$$

A significance of this reformulation is that it is impossible to get a negative *ID*. Thus, equations (3) an (4) provide a formulation that is both more elegant and which provides a better fit with empirical data. We would argue, therefore, that these are the formulations that should be used in future studies.

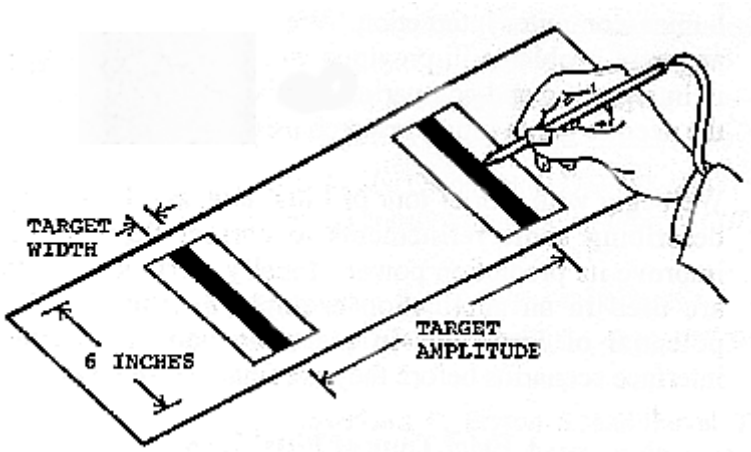

Figure 2: Fitts' Reciprocal Tapping Task
Subjects tapped between the two bars as quickly as possible. The width of the bars and the distance between them were experimental variables. (From MacKenzie, 1991).

The Index of Performance: IP

Fitts' work was largely motivated by an attempt to understand the capacity of the human motor system. Undertaken in an intellectual climate dominated by information theory, he chose to do so by characterizing such capacity as the bandwidth of information articulated by a particular set of limbs. He expressed this capacity as the *Index of Performance*, or *IP,* where:

$$IP = ID / MT \qquad (5)$$

IP is expressed as "bits per second," bits being chosen because of the (arbitrary) base 2 used in formulae (1), (2), (3) and (4)[1]. The higher the *IP,* the higher the human performance since more information is being articulated per unit time.

IP is a useful concept for at least two reasons. First, it provides a metric for quantifying and comparing the capacity of various limbs (fingers, wrist, arms, ...) in the performance of various tasks. Hence, a method of expressing human potential is provided. Second, it provides a way of

[1] Note that *IP* reduces to 1/*b* from equation (1) when the intercept *a* is zero.

quantifying actual performance so that it is possible to compare how well different techniques (such as using different input devices) realize this potential.

Normalization and the Speed/Accuracy Tradeoff

One of the objectives of science is that new work build upon or assimilate that which preceded it; however, while there is a large body of Fitts-based studies, there are real problems in comparing results. One reason is that the paradigm involves both speed and accuracy.

Even at the intuitive level, one can see that there is a trade-off between these two factors. As the old saying goes, "Haste makes waste ..." The more accurate the performance, the longer the movement time, and *vice versa.* Cross-study comparisons of movement times for devices, for example, have little validity unless error rates are the same or otherwise adjusted.

MacKenzie (1991 & 1992) discusses an approach to achieving such normalization. This was first demonstrated by Crossman in 1960 (see Welford, 1968, p. 148), but has not been reflected in published studies. Our objective in presenting it here is to advocate its use as standard practice in future studies.

The normalization technique bases calculations on what subjects actually do, rather than what they were instructed to do. In practice, this means adjusting the target width *W* to be in accord with where user interactions actually took place. Thus, at the model building stage, *W* becomes a dependent variable. We will call this adjusted "effective" target width W_e.

The key to normalization lies in adopting a common criterion for determining W_e. The method advocated by MacKenzie is to use a constant error factor to determine the effective target width. Specifically, he argues the theoretical basis for adjusting target width so as to maintain a constant 4% error rate. That is, based on the coordinates of the experimental data, the boundaries of *We* are determined by the inner limits of the upper and lower 2% of the data, the normalized overshoots and undershoots, respectively. This is illustrated in Fig. 2.

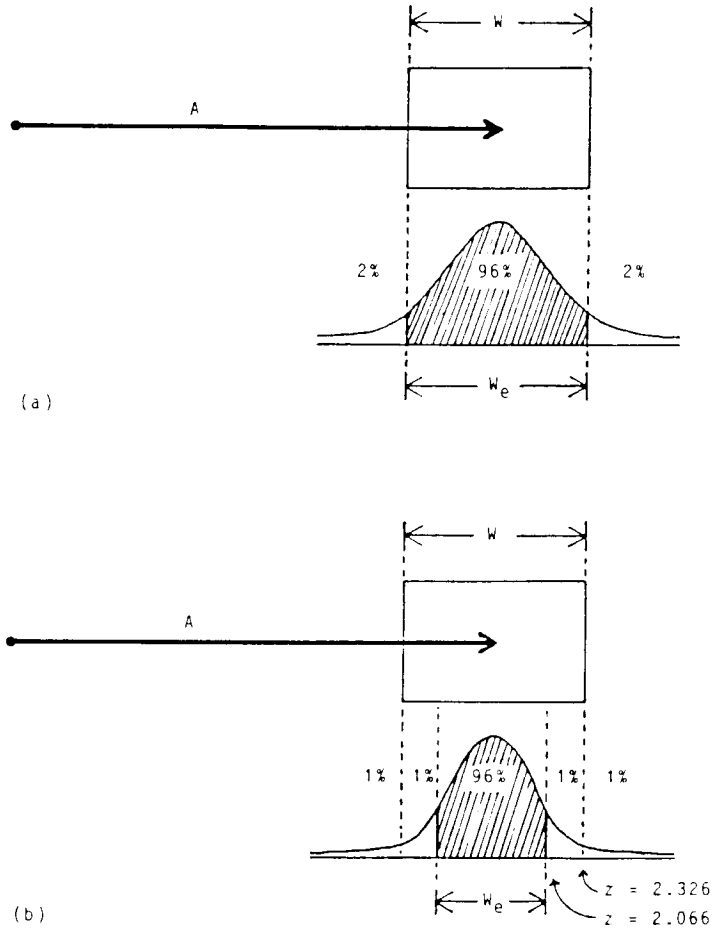

Figure 3: Normalizing Target Width

Target width (W) is adjusted to an effective target width (We) to maintain a consistent error rate of 4%. In (a) the error rate is 4% so no adjustment is required: W = We. In (b) the errors were 2%, so the effective target is narrower than the actual target, in order to raise the error to the normalized value (from MacKenzie, 1991)

That such normalization has important implications has been shown in the first experiment discussed in MacKenzie (1991). This study performed a comparison of three input devices (mouse, trackball and stylus with tablet) in both a target acquisition and a dragging task. Summary data for the index of performance are shown in Figure 4.

Data are plotted using both *W* and *We*. The most important point to note is that the rank ordering of devices changes when normalization is used. Also, values are lower overall using normalized data and the trackball-dragging condition fell from 3 bits/sec to 1.5 bits/sec after normalizing.

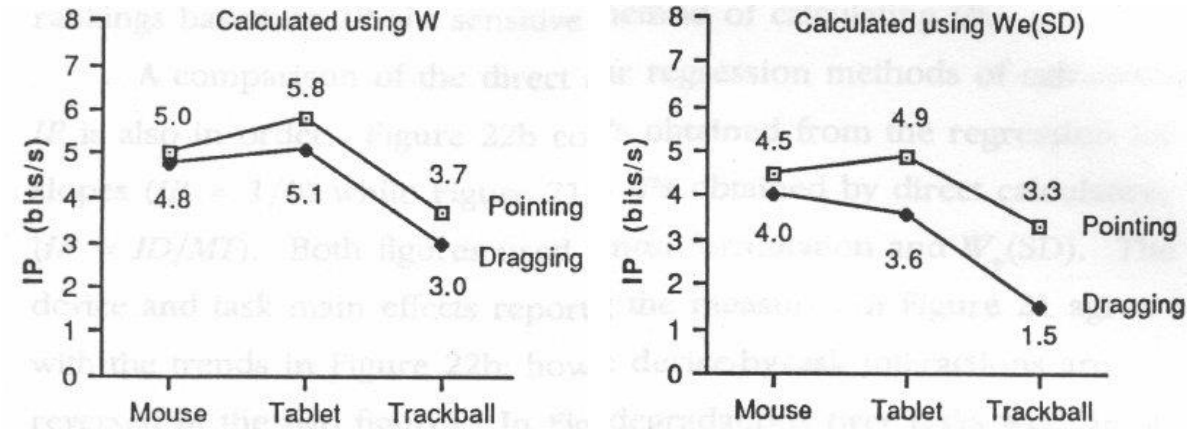

Figure 4: Summary data from Experiment 1 in MacKenzie (1991)
Note the effect of normalizing the target width at 4% errors. The rank ordering of devices in the dragging task has changed and the IP for all devices has been reduced.

These changes are significant. First, if the normalized technique was not used, misleading conclusions would result. Second, normalization (or lack thereof) affects our ability to make meaningful cross-study comparisons. For example, in MacKenzie's (1992) review of six Fitts' Law studies, error rates varied from 0% to 25%. With such a spread along the speed-accuracy continuum, how can reliable comparisons be made? Normalization would eliminate this problem.

Extension to Two Dimensions

It is important to remember that the experiments undertaken by Fitts tested performance between two horizontally separated targets: an inherently one-dimensional task. This is in contrast to the target acquisition tasks that characterize contemporary Direct Manipulation interfaces, which are two dimensional, or three-dimensional interfaces such as those discussed in later chapters.

Some studies, such as Card, English and Burr (1978) and Jagacinski and Monk (1985) have applied Fitts' paradigm to elemental two-dimensional pointing tasks; however, when movements are 2D, they (along with most researchers) applied the model in the usual (i.e., 1D) way: target amplitude is the distance to the target center, and target width is the horizontal extent of the target.

Extending Fitts' Law to two dimensions introduces two new factors that must be properly accounted for:

- How does the angle of approach affect performance?
- What is the width of the target when it is asymmetric, and approached from different angles?

The first issue is rooted in the question of whether we have uniform facility of motion in all directions, or whether there is a bias that results in moving with significantly more efficiency in some directions (left or right) compared to others (such as up and down)? Clearly, such biases can be imposed by the affordances of the transducer used. Just think of the bias towards horizontal and vertical motion (as opposed to diagonal) imposed by the *Etch-a-Sketch* toy.

Such limitations are important and can be exploited in design tasks (or impede human performance when ignored). If our objective is to achieve optimal human performance, then it is important to go beyond the transducer and understand the biases in the human motor system. There are no simple answers, however. Contrast the wrist with the finger or leg, for example: the wrist has more freedom in two dimensions than the latter two. Any application of Fitts' Law in two (or more) dimensions that ignores this variation in the human motor system is misguided, and will result in misleading conclusions.

There are, however, studies that do provide data to support the Fitts paradigm to model two-dimensional pointing tasks. In particular, Card, English and Burr (1978) found that approach angle had little significance. In comparing movement time along the diagonal axes to the vertical or horizontal axes, they found no difference for the mouse, and only a 3% increase along the diagonal axes for the joystick. In keeping with the preceding comments, however, the reader is strongly cautioned from generalizing from these results without further experimental validation or careful analysis.

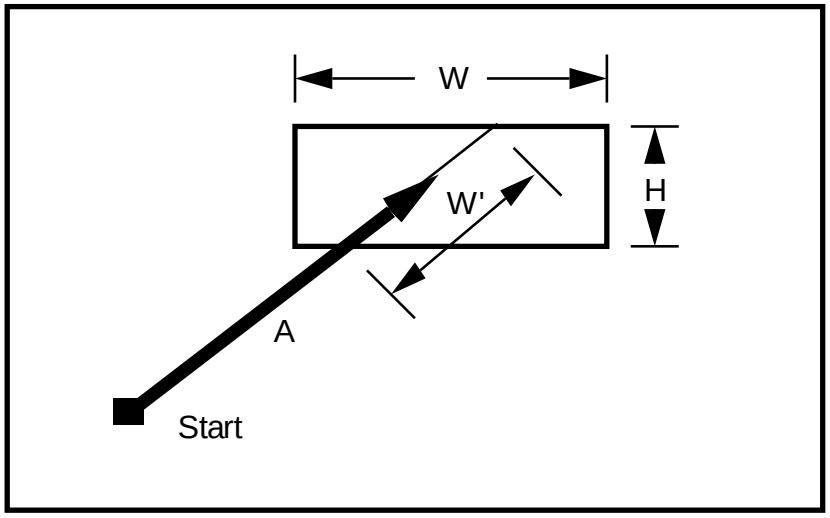

Figure 5: Determining Effective Target Width

In acquiring a two-dimensional target, how is the target width term calculated? Approaching from the angle indicated, is it the height (H), width (W), diagonal (W'), or something else? (from MacKenzie, 1991)

The second issue that arises in applying Fitts' Law in two dimensions is how to determine the target width, when the shape is asymmetrical and the target is approached from different angles. Consider, for example, selecting a word in a document processor. The target height is determined by the height of the font used, and the target width by the font size and the number of characters in the word. Assuming that the average word has five characters and that characters are more-or-less square, the target width is typically five times the height. So which value should we use for the *W* term in equations (1) or (3): height or width? Should we always use the same value, or, in approaching from the side should we use the width, and from above or below, should we use the height?

Using a more graphical example, consider acquiring the target shown in Figure 5 from the indicated angle of approach. Should the *W* term be the target's height, width or diagonal?

A number of strategies have been discussed to determine what value to use for the *W term*. Gillan, Holden, Adam, Rudisill and Magee (1990), for example, investigated the following alternatives:

- H: target height
- W: the width of the target
- H + W: target height plus width
- H x W: target area - height times width

The results of this study are problematic, however, since how they defined the underlying task did not closely conform to the Fitts paradigm. Consequently, the question was investigated further by MacKenzie (1991) who examined the following heuristics for calculating effective target width:

- W: target width

- W + H: target width plus height
- W x H: target area - height times width
- SOWH: the shorter of width or height
- W': projecting the approach vector through the target, the length of the portion intersecting the target (shown as W' in Figure 5Figure 5)

The models evaluated were chosen partially based upon what models had been used in previous studies. From the results of this study, the W + H and W x H models were discarded as being no better than the status quo, W. The SOWH and W' models were both superior to the W model, and did not differ significantly from one another.

Consequently, based on this study, where applicable, the SOWH is the preferred model, since it is computationally much more efficient than the W' model. However, the W' model is still the superior of the two in that, unlike the SOWH model, it works for targets that are non-rectangular, and it preserves the inherent one dimensionality of the Fitts model.

Repetitive vs Discrete Tasks

Fitts' original task was a reciprocal tapping task. The repetition in the task was key to Fitts' paradigm. Since the task was repetitive, the planning of actions consumed no load; rather, the focus was on the motor action of articulation.

This repetition is not typical of most real-world target acquisition tasks, such as those encountered in using Direct Manipulation interfaces. These are discrete tasks, done once, and preceded and followed by some other task.

Data for discrete tasks are different from that for repetitive tasks. The difference is a higher *IP* for the discrete case (Fitts & Peterson, 1964). This difference is likely due partially to the time consumed in planning the action, such as searching for the target. In applying results from the literature, it is important to take this difference between discrete and repetitive tasks into account.

Extension to Dragging Tasks

- value of Fitts' Law is its power as analytical and predictive tool
- modeling human performance with such can be very useful in design process
- problem is small set of transactions found in typical systems which can be modeled in such a way
- Fitts' Law works well for target acquisition tasks, and there are models for less relevant pursuit tracking tasks, but what about tasks such as dragging or inking?
- an important recent development has been to show that Fitts' Law can be extended to modeling dragging tasks
- examples of applying Fitts' Law to dragging are Gillan *et al* (1990), and MacKenzie, Sellen & Buxton (1991)
- appeal is that a known and established model can be applied, and results can be related to other tasks (viz., target acquisition) modeled using the same paradigm
- that dragging can be considered a Fitts' Law task is seen intuitively in the following example, illustrated in Figure 6.

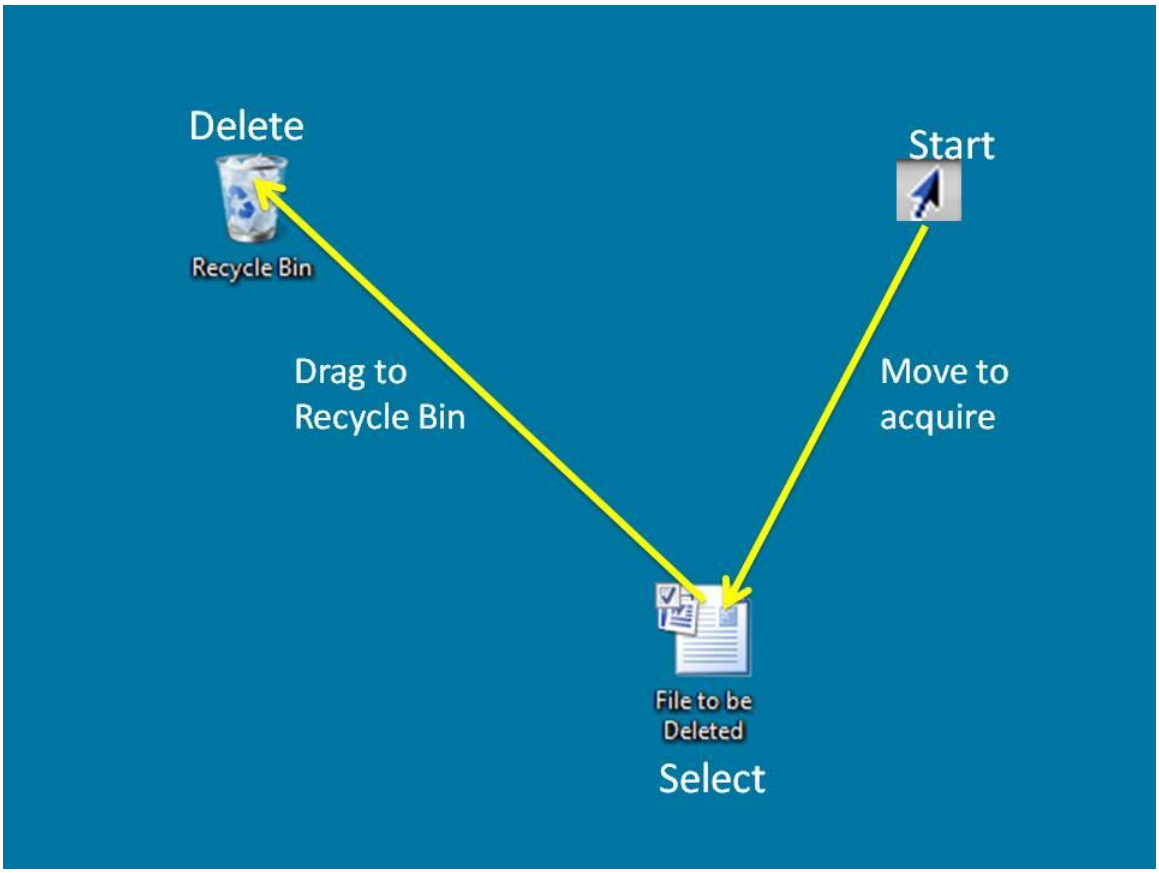

Figure 6: Deletion as Compound Fitts Task

Deleting an icon on a PC can be considered two Fitts' Law tasks. The first is a standard target acquisition task. As illustrated, this is composed of moving from the start position and acquiring the icon of the file. The second is the acquisition of the trash icon, starting from the position of where the file to be deleted is acquired. The main difference between the two tasks is whether the mouse button is up or down during the task performance.

The example illustrates deleting a Word document by dragging it to the Recycle Bin. Performing this transaction involves two main steps:

- Acquiring the icon representing the file to be deleted
- Dropping it into the Recycle Bin.

Each of these sub-tasks is a target acquisition task that can be modeled using Fitts' Law. In each case, one moves amplitude *A* from a start position to a target of width *W.* However, there is a difference between the two which can be expressed using the language of the 3-State Model introduced in Chapter 4: the first is a State-0 action whereas the second dragging task is State-1.

From the motor-action perspective, when performing these tasks with a mouse and a conventional GUI, for example, the significance of this difference is that the user must hold down a mouse button while performing the second (dragging) task. This is an important observation and an example of the vocabulary that we are developing and these models are useful in analysis and design. What these models and vocabulary help us see is the following:

- State-2 tasks are common in graphical user interfaces
- Nearly all comparative evaluations of input devices test the State-1 case only.
- In the State-2 case, the user frequently (usually) has to hold down a button while performing the Fitts Law task.

- Doing so can interfere with performing the Fitts Law task.
- This is shown in MacKenzie, Sellen and Buxton (1991).
- The degree of the impact varies with the device used (mouse, trackball, joystick, touchpad, etc.) due to the degree of interference imposed.

Consequently, in designing interfaces, one must consider the interaction technique and the device used together, since they are mutually dependent. Furthermore, one needs to make sure that one uses the parameters appropriate to the State of the task (1 or 2) and the device used, when modeling the task. .

Fitts' Expt. 3: The "Prince" Technique

Forgotten part

Zhai & Buxton

Relate to Toolglass, etc.

Practical Application: an Example

To illustrate how Fitts' Law can be practically used in the design process, we will work through an example from last chapter of MacKenzie (1991). The example uses models derived from Fitts' Law and experimental data to compare three different ways to delete a file on the Apple Macintosh. The techniques considered are:

Point at and select the icon of the file to be deleted and drag the icon to trashcan. This is the traditional method used with the Macintosh. It is a compound task. The second dragging portion is a State-2 Fitts task.

Point at and select the icon of the file to be deleted, then point at and select the trashcan. This is a variation on the traditional method. In this method, both sub-tasks are State-1 Fitts tasks.

Draw a left-to-right stroke through the icon to be deleted. This is a technique uncommon with most GUIs, but typical of the methods of interaction found in the emerging pen-based "mark-up" interfaces (discussed in Chapter 13). The technique is illustrated in Figure 7.

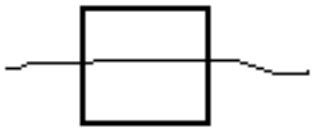

Figure 7: Deleting an object by drawing a stroke through it.
The figure illustrates deletion by drawing a stroke through the icon representing the object to be deleted. In this case, the icon is represented as a square (after Kurtenbach & Buxton, 1991).

Figure 8 is a representation of this task in a form that enables us to model it using the techniques discussed in this chapter. Here we have two icons. These are shown as squares draw with bold lines. One represents the file to be deleted and the other the trashcan. For the example, we make the following assumptions:

- the icons are 2 cm square
- the distance between the two icons is 14 cm
- the task is performed using the Macintosh mouse

Since all three methods being compared require the user to acquire the icon of the file to be deleted, we will not consider that part of the task further. We will assume that the pointer is either over the file icon (methods 1 and 2), or in the appropriate position to begin drawing the delete stroke (method 3).

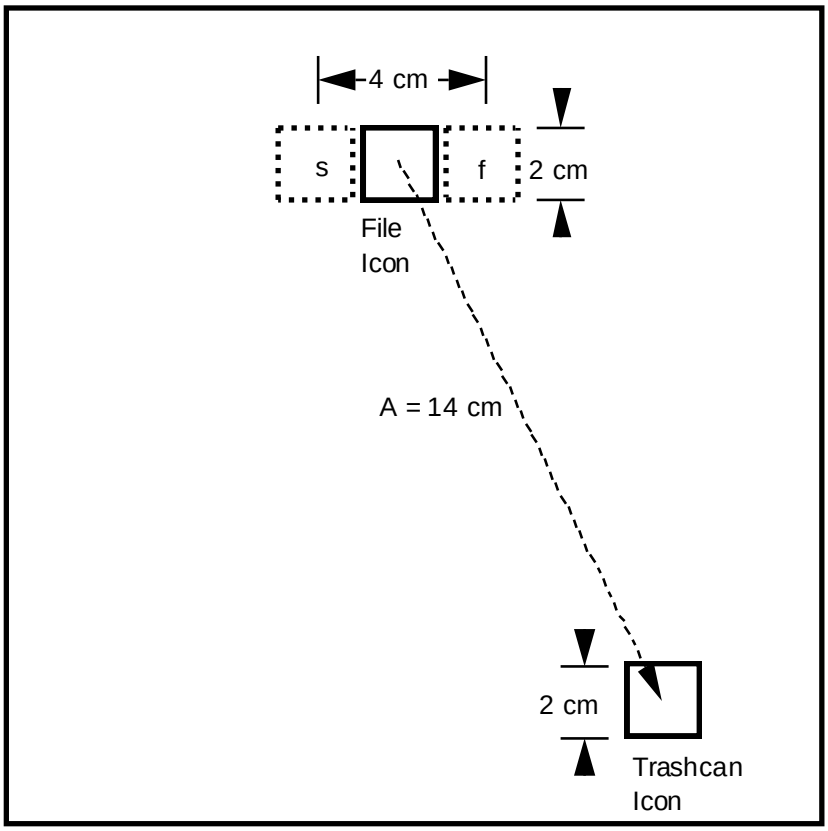

Figure 8: Modeling Methods for Icon Deletion

On the Apple Macintosh computer, icons (represented by solid line squares) are deleted by dragging them to the trashcan. In the example, this could be modeled with Fitts' Law, using (as an example) A=14cm and W=2cm. Alternatively, the icon could be deleted by drawing a stroke through it. This we could model as drawing a line starting from the centre of an imaginary box "s", and finishing in an imaginary box "f" ("s" and "f" shown as dashed boxes). In this case, A=4cm, and W=2cm. (From MacKenzie, 1991).

We will use equations based on experimentally derived data in our calculations. First, State-1 *MT* using the Macintosh mouse will be calculated as (MacKenzie, 1991, eqn. 26, p. 102):

$$MT = 230 + 166\ ID \qquad (6)$$

State-2 *MT* (i.e., dragging), will be calculated as (MacKenzie, 1991, eqn. 25, p. 94):

$$MT = 135 + 249\ ID \qquad (7)$$

Modeling Method 1 - Deletion by Dragging to Trashcan:

$MT = 135 + 249\ ID$ *from (7)*

$= 135 + 249 \times log_2 (A/W + 1)$ *from (4)*

$= 135 + 249 \times log_2 (14cm/2cm + 1)$

$= 135 + 249 \times 3$

$= 882\ ms$

Modeling Method 2 - Deletion by Selecting Trashcan**:**

MT = 230 + 166 ID *from (6)*
= 230 + 166 x l o g_2 (A/ W + 1) *from (4)*
= 230 + 166 x l o g_2 (14cm/ 2cm + 1)
= 230 + 166 x 3
= 728 ms

Modeling Method 3 - Deletion by Stroke-Through**:**

We will model the task using Fitts' Law by assuming that it involves drawing a line from the centre of a virtual target on the left of the icon to the centre of another virtual target on the right of the icon. This is shown in Fig. 7, where the virtual targets are represented using dashed lines. We will assume that the virtual targets are the same size as the icons. Therefore, A in this case will be 4 cm, and W will be 2 cm.[1] Since drawing the stroke involves a State-2 task, we will use equation (7) in our analysis.

MT = 135 + 249 ID *from (7)*
= 135 + 249 x l o g_2 (A/ W + 1) *from (4)*
= 135 + 249 x l o g_2 (4cm/ 2cm + 1)
= 135 + 249 x 1.58
= 528 ms

There are some interesting results that come out of this simple analysis. First, in terms of movement time, the actual method used by the Macintosh performs the worst. Second, but more subtle, all other things being equal, notice that if the size of the display changes (for example, becomes much larger), movement time using Method 3 is a constant, while it greatly increases for Methods (1) and (2) as the distance between the icon and the trashcan grows. (This is familiar to Macintosh users who have moved from the original 9" monitor to the newer large monitor systems.)

This is a good point to make a cautionary note, however; movement time is not the only measure to consider. As an exercise, after reading the section on *Chunking and Phrasing* later in this chapter, come back and reanalyze these three techniques in terms of proneness to error based on the "phrasing" of their articulation.

Summary

- use new formulation (3)
- normalize target width
- use appropriate approximation of width in 2D
- model applies to dragging
- evaluate deices in both State-1 and State-2 Tasks
- develop models of other representative transactions
- evaluate devices w.r.t. each task, and weight according to predicted frequency of occurrence in target usage.
- consider real-world context in creating models

[1] While this is not an ideal method of modeling the task, it is acceptable for our purposes. If anything, results obtained will be poorer than what would occur in practice, since the real transaction does not have the constraint of starting or ending within the virtual targets. Any error due to this method of analysis works against the technique. Therefore, if the technique out performs the others, the analysis is all the more convincing.

  - real tasks normally discrete, not repetitive
  - Fitts' Law not sufficient in itself
  - real situation compounded typically by search time for target and planing time
  - in cases where limb is not in "home position" on the pointing device, the time to acquire transducer must be considered
- homing time may be more significant than pointing time
  - different devices have different homing times. e.g., stylus much harder to acquire from keyboard than trackball, yet performance is just the opposite in pointing time.
  - leads to higher level models, such as Keystroke Level Model, discussed later in this chapter.
  - need to extend to higher dimensions (e.g., Ware & Osborne, 1990) but following the above considerations
- *Incomplete.* See the following additional references for further information:
  - Boritz (1990). for discussion of handedness and approach angle.
  - Hoffmann, E.R. (1991) for moving targets

Two Handed: **MUST INCLUDE**:

Mottet, D., Guiard, Y., Ferrand, T. & Bootsma, R. (2001). Two-Handed Performance of a Ryhthmical Fitts Task by Individuals and Dyads. *Journal of Experiemntal Psychology: Human Perception and Performance,* 27(6), 1275-1286.

The Steering Law

As we can see, the simple and robust human movement regularity modeled by Fitts' law serves as a very powerful tool for user interface research and design. However, Fitts' law models a specific task paradigm, the Fitts tapping task, which corresponds to pointing task on a computer screen. There are many HCI tasks that do not match the paradigm. One class of them is steering: moving along trajectories on a computer screen. Navigating through nested-menus and drawing curves are two examples of steering task (Figure 9).

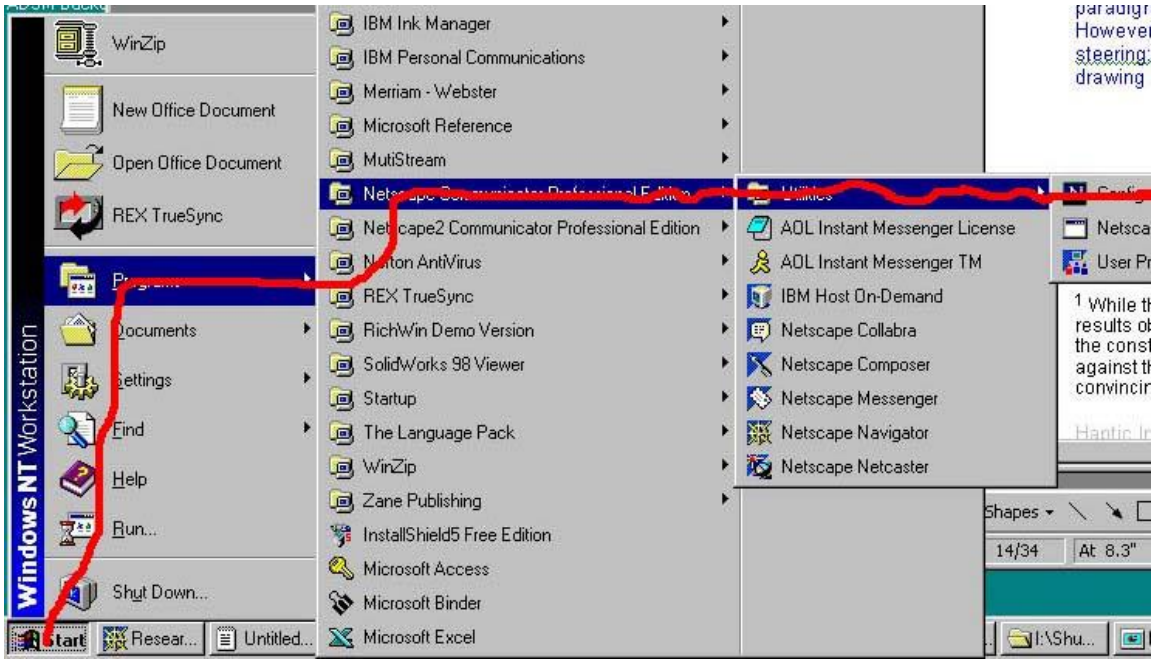

Figure 9: Traversing nested menus involves multiple segments of steering tasks

Obviously, due to the different nature of steering, one can not expect the movement time in steering follow the Fitts' law equation. But is there a similar regularity in steering? Recently, Accot and Zhai (1997) discovered that a lawful relationship indeed existed between steering task difficulty and human steering speed. Furthermore, their "Steering law" could be linked with Fitts' law.

Accot and Zhai first confirmed that passing through "goals", as illustrated in Figure 10, also follows Fitts law. In this task, the accuracy constraint (W) is perpendicular to the cursor's movement direction. This is opposite to Fitts' tapping task, which requires accuracy along the movement direction. Furthermore, Fitts' tapping task requires the cursor to land (stop) on the target, while the goal passing task only requires the user to pass the goals that has certain width. Nonetheless, Accot and Zhai showed that this types of task follows the same Fitts' law speed accuracy relationship:

$$MT = a + b \; l o g_2 (A/W+1) \qquad (ST1)$$

where MT is the time duration between passing the two goals; A is the distance between the two goals and W is the width of each goal. Note that this finding has application values in itself. For example, in the desktop operating interfaces, one often needs to activate pull down menus at the very edge of the computer screen, such as the File menu in Macintosh, and Start menu in Windows. In some cases (such as the Apple Macintosh), the pull down menu button is virtually beyond the screen. Activating such a button hence can become a goal passing task: simply pushing the cursor through the File button and it would be stopped on the button. Users can take advantage of such an effect to avoid a more difficult Fitts' tapping task. The amount of time saving by employing such a technique can be computed given that goal passing also follows Fitts' law.

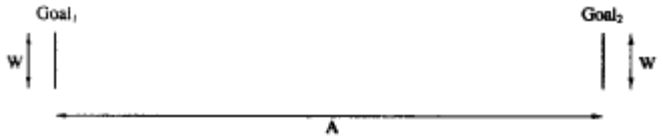

Figure 10: A two goal passing task also follows Fitts' law

Using the goal passing task as a stepping stone, Accot and Zhai conducted a thought experiment that placed N number of goals along the movement trajectory and mathematically proved that the new index of difficult would change from

$$log_2 (A/W+1)$$

to

$$N \; log_2 (A/NW+1)$$

Taking the number of goals to infinity, the continuous goal passing task would become a steering task (Figure 11, we can also image the thought experiment as chopping the steering tunnel into an infinite number of goal passing tasks) with Index of Difficulty:

$$A / W \; ln(2)$$

Combing the constant *ln(2)* with *b*, we have

$$MT = a + b \; A/W \qquad (ST2)$$

This means that the time to steer through a tunnel as shown in Figure ST3 without ever hitting the boundaries is proportional to an index of steering difficulty. In the case of a straight tunnel, the steering ID is A/W. An experiment, under a varied set of ID (A = 250, 500, 750, 1000 pixels; W = 20, 30, 40, 50, 60, 70, 80, 90 pixels), showed that data collected with actual human subjects strongly correlated with Equation ST2 (r = 0.968).

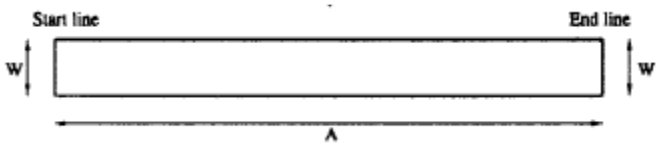

Figure 11. Steering through a tunnel follows the Steering Law

It was further shown that such a lawful relationship also existed for tunnels with different shapes, such as a cone or a spiral shape, all experimentally verified at greater than 0.96 fitness. For an arbitrary shaped tunnel C, as illustrated in Figure 12, the most generalized form of steering law is:

$$MT = a + b \int_C 1/W(s)\ ds \qquad (ST3)$$

In other words, the movement time to steer through C is proportional to its index of difficulty, which is the integral of the inverse of the width along the path.

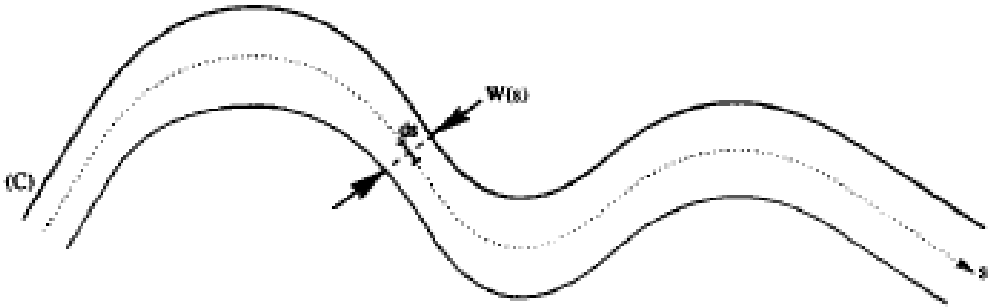

Figure 12. Generalized steering task

Similar to Fitts' law, *a* and *b* are constants dependent on the shape of the tunnel, the device used in the task and the individual who performs the task. For the same steering tunnel shape and the same group of subjects, *b* can be used to measure the control quality of an input device. See Accot and Zhai (1999) for an example of applying the Steering law to evaluate input devices.

In summary, as a developing discipline, research in the field of HCI tends to be "soft". Many workers, such as A. Newell and S.K. Card (1985), have argued that the advancement of HCI lies in "hardening" the field with quantitative and robust models. The steering law is one such an effort. It carried the spirit of Fitts' law a step forward and explored the possible existence of other robust regularities in interaction tasks. It showed that there is a simple linear relationship between movement time and the ``tunnel'' width in steering tasks. The regularities presented in the steering law may enrich the small repertoire of quantitative tools in HCI research and design.

Control:Display (C:D) Ratio

The C:D ratio[1] is the ratio between the degree of control asserted by the user and the degree of response that the system exhibits as a result. For linear controls, such as controlling the movement of an arrow cursor on a computer display by moving a puck on a digitizing tablet, the C:D ratio would be the ratio between the distance the user moves the puck (the control, C) and the resulting distance that arrow cursor moves on the computer screen (the display, D)[2]. For example, if the C:D ratio was 2:1, moving the puck two centimeters on the tablet would result in the pointer on the display moving only one centimeter.

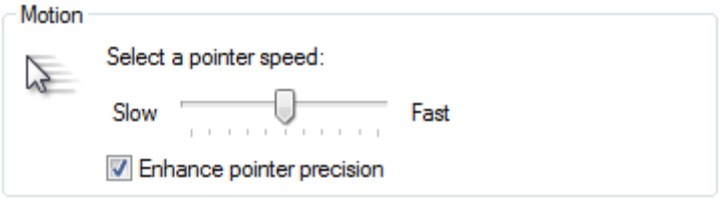

Figure 13: Pointer Speed and C:D Ratio
When you adjust the pointer speed for your mouse, as with a widget such as that shown above, you are changing the C:D ratio.

If you have ever adjusted the sensitivity of your mouse using a widget like that shown in Figure 13, then you have had some experience adjusting the C:D ratio on your computer. In this section we are going to probe deeper into this topic since an understanding of C:D ratio is one of the tools that can help us find a better balance amongst input device, task, and user in the design of interactive systems.

As a start, and to keep our focus on the user, we need to introduce a few basic concepts pertaining to human motor control. Consider the following two actions:

1. Throwing a ball as far as you can without caring where it lands.
2. Tracing the image on a postage stamp as accurately as you can.

These two tasks represent opposite extremes. While they may appear to have little to do with each other, on closer examination we will see that much of what we do in interacting with computers involves a combination of the two. For example, imagine that you are editing a document on a very large display, and your pointer is in the bottom left-hand corner of the screen. Now imagine that for some reason you have to use your mouse to select the superscript of the following number "2^n", which is in the upper right hand corner of the same display.

Taking a page out of our earlier discussion of time-motion analysis, the motor action that would typically be used to perform this task into two steps that correlate very closely to the two actions discussed above:

1. Quickly get over to roughly the top right corner of the screen, without worrying too much about acquiring the target.
2. Once nearly there, carefully selecting the superscript.

[1] In the literature, C:D ratio has also been discussed under the terms C/D ratio, control-display gain, and display/control gain,

[2] This is a topic that has been covered in great detail in the human-factors engineering literature. For a summary, the interested reader is referred to Van Cott & Kinkade (1972).

In the literature there are some terms that have been used to characterize these two types of action[1]. These include:

- *Gross vs Fine*: This characterization typically has to do with the scale of the motion and/or the muscle groups used. Both throwing the ball and moving across the screen typically involve a larger range of motion than tracing an image or selecting a small target. Likewise, moving across a large screen with a mouse would tend to involve movement in the fore-arm and wrist - and the associated larger muscle groups - compared to selecting the subscript, which would more likely involve the fingers and wrist.

- *Ballistic vs Regulated*: Throwing the ball or moving quickly across a large display are examples of Ballistic action. This can be thought of as issuing a command to the intended limbs and muscle groups, and then letting them go off and do what they were told. Overall control of the end result is mainly determined in the initiation of the action, not during it. This is contrast to the tracing or superscript selection tasks, where there is constant monitoring and consequent adjustment during task execution.

- *Open Loop vs Closed Loop*: Another way of characterizing these two types of motor action is that the latter has a much tighter feedback loop between motor-action and perception of the current state during execution. Open Loop states that there is little or no opportunity to make adjustment during the action. As already mentioned, once initiated, the die is cast. On the other hand, closed loop behaviour implies that there is ongoing opportunity for adjustment, based on feedback.

Remembering that both types of motor action frequently occur in performing the same task, two things that we need to keep in mind as designers are (a) that the optimal C:D ratio is different for the two types of task, and (b) the choice of input device can have significant implications in terms of our options around C:D ratio.[2]

To get a sense of the former, let's dive back into the literature to a study published by Jenkins and Conner (1949). They had their subjects perform a task that was broken down into two phases, one which required coarse movement along a dimension, and the other fine adjustment. The data from their study is shown in Figure 14 and their associated observation was:

> For all subjects, travel time declines rapidly with increasing coarseness to about 1.18; thereafter coarser ratios do not speed up travel materially. In the opposite fashion, adjusting time declines with *decreasing* coarseness of ratio to about 1.18; thereafter finer ratios do not aid in making the final adjustment. A ratio about 1.18 combines rapidity of travel with speed of final adjustment. (Jenkins & Connor, 1949, p. 400)

If one assumes that the occurrence and importance of the two types of motor action are about equal, then the optimal C:D ratio for this situation would be as indicated in the figure: where the two performance curves intercept. The other thing that their comments should keep you mindful of is that the improvements achieved in adjusting the C:D ratio are bounded. More than enough of a good thing does not bring further improvement.

[1] In this chapter we introduce the basic concepts. In the next chapter, Human Performance, we will go into more detail.

[2] In making these distinctions I am not suggesting that all motor action falls into one category or the other. Think about these concepts is as two extremes of a continuum. The value lies in teasing out attributes of human action that can help inform our design decisions.

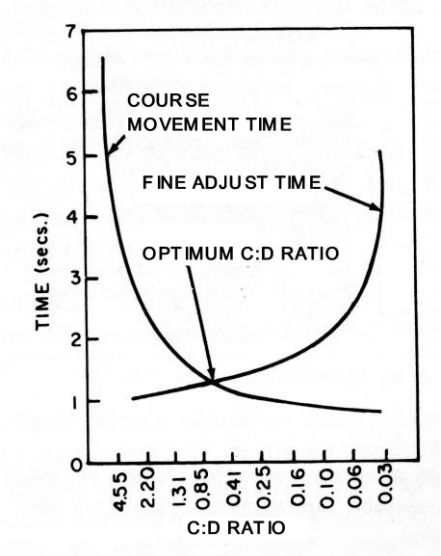

Figure 14: C:D Ratio and Motor Action Type

The C:D ratio associated with the lowest time for the course (ballistic) movement results in the longest time for the fine motor action, and vice versa. The optimal C:D ratio for this task is at the intercept of the two. (Adapted from Chapanis & Kinkade, 1972, which was adapted from Jenkins & Connor, 1949).

Earlier in Chapter 3, Alternative Perspectives, we made a distinction between direct and in-direct devices (e.g., touch screen *vs* touch tablet) as well as motion and position sensing devices (e.g., mouse *vs* tablet puck). Building on this, in our discussion of device taxonomies in Chapter 4, we saw that devices could be characterized by the human potential that they sensed, such as position, motion and pressure. We can now delve in a bit deeper and see how these properties interact with C:D ratio in terms of affecting what they may or may not be suited for in various circumstances.

Direct Linear Controllers

For a start, let's take direct position sensing devices such as touch screens or stylus-driven tablets, where the on-screen tracking symbol (such as the arrow cursor) directly tracks the position of the finger or stylus on the screen. Inherently, such devices have a 1:1 C:D ratio.[1]

Keep in mind, however, that C:D ratio is just that, a ratio. The actual range of motor action required in operating a direct position sensing device is determined by the size of the display. For example, consider such an interface on each of the following scales: a wrist watch, a mobile phone, a PDA, a Tablet-PC, and electronic whiteboard. From the extreme conditions, we see that despite having the same 1:1 C:D ratio, our ability to interact with the display is constrained by the limitations of our acuity of fine motor control on the one hand, and the limits of our reach on the other.

Indirect Linear Controllers

Now let's look at indirect position sensing devices, such as graphics tablets controlled by a puck or stylus, where the on-screen pointer directly tracks the position of the puck or stylus on the tablet surface. For such devices, the C:D ratio will again be *fixed*, but the actual ratio will be determined by the ratio between the size of the tablet surface used for control, and the size of the

[1] One consequence is that this may (but may not) render the need for an explicit tracking symbol redundant. As with traditional media, your finger or the stylus may be all that is needed to indicate where you are pointing.

associated display. For example, if the tablet dimensions are double that of the display, the ratio will be 2:1, if they are the same, 1:1, and if half those of the display, 1:2. An actual example is illustrated in Figure 15.

Figure 15: A Fixed C:D Ratio Indirect Device

This is the Bamboo tablet from Wacom. Moving the stylus on the tablet surface controls the movement of the pointing symbol on the associated computer display. The actual C:D ratio is a function of the dimensions of the active area of the tablet (147mm x 91mm) compared to those of the display. Assuming the LCD of my laptop is 260mm x 161mm, the effective C:D ration using this tablet on my laptop would be 1: 260/147 = 1:1.77. However, if I used the same tablet on my desktop computer, whose display is 470mm wide, the C:D ratio would be 1: 470/147 = 1:3.2. (Photo: Wacom Co.)

As the data shown in Figure 14 indicates, the best that we can do with fixed C:D ratio devices is find the best compromise between optimizing for ballistic action and fine control. With direct position-sensing devices, the C:D ratio is fixed at 1:1, so our only real option is our choice of display size. With indirect position-sensitive devices, we can control the range of motor action by our choice of size of the control surface, and we can control the C:D ratio by our choice of display size, relative to that of the control surface.[1]

Non-linear Indirect Devices

In contrast, motion-sensitive devices such as the mouse and the trackball offer a far more flexible approach to accommodating the otherwise conflicting demands of ballistic *vs* fine motor control. While such devices can be set up to provide linear control (what is frequently, but strictly

[1] One trick to achieve even more dynamic control over the C:D ratio of such devices is to map only a portion of the tablet surface to the screen. For example, assuming you start with a 1:1 C:D ratio, using just one quadrant of the display would change the C:D ration to 1:2. Some applications enable one to dynamically change the size of the active control surface. For example, one might use only a portion of the surface for basic pointing tasks, such as selecting icons or accessing menu items, but using the whole surface for high precision tasks such as drawing.

speaking, incorrectly, called “tablet mode”), this is rarely done. Rather, with these devices, the C:D ratio is typically set to change dynamically depending on the speed at which the device moves. Here is a simple exercise that I encourage you to actually try, rather than just read about:

1. With your mouse, position the tracking arrow of your computer on the left side of your display.
2. Mark the position where your mouse is on your desk top. This is your starting point.
3. Move your mouse very slowly, in a continuous motion, to the right.
4. Stop when the tracking arrow reaches the right side of the display.
5. Again, mark where the position where your mouse is. This is the slow finishing point.
6. Move the tracking arrow back to the left side of the display.
7. Pick up your mouse and place it at the starting point that you previously marked.
8. Repeat steps 3-7, this time moving at a moderate speed. In this case, in step 5, you will be marking the medium finishing point.
9. Repeat steps 3-5, this time moving as fast as you can – but be careful not to over-shoot with the mouse. Stop as fast as you can as soon as the tracking arrow reaches the right side of the display. The end position of the mouse is the fast finishing point.

What you will see is that the slower you go, the further you have to move the mouse in order to get to the far side of the screen. Now go to the control panel on your computer that controls the mouse behaviour, and with the panel similar to that in Figure 13,set the mouse response to “slow”. This gives you a high C:D ratio, i.e., you have to move the mouse much more to cause the arrow pointer to move a given distance on the screen. Repeat the exercise above, and you will typically see even a greater difference amongst the three end positions.

Most people are unaware that with most graphical user interfaces that the C:D ratio varies according to the speed at which the mouse is moved. That they don’t is a pretty good indicator that the designer has got it right. Of course, this comes at a price. If one tries to trace an image using a mouse, the non-linearity of the mouse’s C:D ratio will be noticed immediately, due to the distortions in the resulting image on the screen, compared to the original.

Again, please don’t take my word for it. Get a line drawing on a sheet of paper, and do it – trace it into a graphics program in your computer. This is a great exercise to illustrate that everything is best for some things and worst for something else. Tracing is best done with linear controls, whereas the non-linear C:D ratio of motion-sensitive relative controls such as a mouse, provide the best accommodation for both ballistic and fine motor action.

From Linear Position-Sensing to Non-Linear Motion-Sending

In the last tracing exercise, if your computer allowed it, you may have tried to improve the result that you got with the mouse by switching from “mouse mode” to “tablet mode”. Insofar as individual lines are concerned, you would see an improvement. The reason is that doing so puts the mouse in linear rather than non-linear mode. Hence, the C:D ratio does not chance, so the path that the mouse traces on the control surface should be accurately mirrored on the display.

The problem, however, is that even though the path is captured, what is sensed is motion, not position, so any path is only correct in a relative, rather than absolute sense. To illustrate what I mean with an example, note that if you want to move the tracking arrow vertically up and down mid-way across the screen, you can do so as long as you move the mouse on the vertical axis – and it doesn’t matter where the mouse is initially positioned left or right, as long as the tracking arrow starts at in the middle of the screen.

What this demonstrates is that a relative controller like a mouse can be linear, but it is not position sensitive. On the other hand, a linear position-sensitive device, such as a puck on a digitizing tablet, can effectively emulate a motion-sensitive device, such as a mouse, in such a way as the tablet surface appears to behave as simply a fancy mouse pad.

Figure 16: What Kind of Device is This?

If your answer was "A mouse," you are wrong. This is the puck for a Wacom Intuos digitizing tablet. (I removed the Wacom label in the photo so as not to give away the answer.) A mouse is a relative device that senses motion on the desktop. But if you lift it up and put it down somewhere else on the desk, and the tracking arrow will not move on the screen. A mouse does not sense, or care, where it is positioned. That is a key affordance that lets it have a non-linear C:D ratio. Despite looking like a mouse, this puck needs to be used on a tablet – which itself looks somewhat like a mouse pad. But, it is the position of the puck on that pad that is sensed. Like the mouse, if you move the puck on the tablet left and right or up and down, the tracking arrow will follow – but it is following the pucks position, not its motion. While you need to move to change position, the two are not the same. You can see this if you pick the puck up and then place it down at a different location on the tablet. Every time you put it in a particular place, it will cause the tracking arrow to be positioned at the same place. This position-to-position mapping is the affordance that dictates a linear C:D ratio. (Photo: Wacom)

The approach is pretty much as simple as it is old: instead of using the absolute position of the puck on the tablet to set the coordinates of the tracking symbol, you take the *difference* between successive coordinates, and use them to control the motion of the device. By pushing down one derivative, the position-sensing device becomes a motion-sensing device.

Now this may seem like an arcane bit of historical trivia, and something that is pretty irrelevant to you in this modern era of pen computing, gesture control, tangible computing and multi-touch. But actually, just the opposite is true. All of this information is relevant fodder for the creative designer. But I will leave you with this thought, for now at least: when I look at people interacting with large multi-touch table-top and wall-mounted displays, much less the tool-kits developed to support such interaction, one of the things that I see most neglected is any consideration of when to use relative *vs* absolute control and varying, including when and how to effectively and dynamically switch from one to the other, and when and how to dynamically adjust C:D ratio. And, in this, note the implied assertion in the statement that one *can* effectively use relative touch control on a direct touch surface and therefore variable C:D ratio. And while I am at it, let me also state that in such cases, it is perfectly reasonable – and natural (when properly implemented) – to

vary such things independently for each hand in bimanual interaction, much less amongst different people, when working collaboratively on the same surface.

No matter how arcane some of this material may seem, buried within are the seeds for future insights and the foundation for solid design decisions.

Still to consider incorporating:

Among other things:

Arnaut, L.Y. & Greenstein, J.S. (1990). Is display/control gain a useful metric for optimizing an interface?, *Human Factors,* 32(6), 651-663.
Buck, L. (1980).
Rutledge, J. & Selker, T. (1990). Force-to-motion functions for pointing. In D. Diaper et al. (Eds), Human-Computer Interaction - INTERACT '90, Elsevier Science Publishers B.V. (North-Holland), 701-706.
Arnaut & Greenstein (1986).
Becker, J. & Greenstein, J.S. (1986)

Really big displays
Working far from display
Higher resolution of displays

Cognition and Human Information Processing

The Issue of Limited Resources

There are two concepts from information processing that are of particular importance in understanding current theories of cognition. These are the notions of:

- critical resources
- limited resources

Critical Resources are resources that are required to perform a particular task, or execute a particular process.

Limited Resources are just what their name suggests, resources that are in short supply relative to their need. In cognition, as in operating systems, problems arise when critical resources are limited, and supply cannot meet the demand. One aspect about the human information processor, when compared to the typical computer, is that performance generally degrades gracefully when resources become so saturated.

The Resource Utilization of Tasks

Various tasks and cognitive activities can be discussed in terms of the resources that they consume: both in type and in quantity. This is something developed, for example, in Norman and Bobrow (1975). In their paper, Norman and Bobrow characterize processes as being limited by either

- the processing resources available, or,
- the quality of data available The issue of available processing

The issue of available processing *resources* has to do with memory, processing ``cycles'', and internal communications channels. If adding additional resources will improve performance, then the task is said to be *resource limited.*

Data quality has to do with what is being processed, rather than what is doing the processing. Once we reach a point where adding processing resources does not improve performance, the process is said to *data limited*. That is, no improvement in performance can be achieved without improving the quality of the data available for processing. Data-limited processes can be further broken down into two categories, those that have:

- signal data-limits, and those that have
- memory data-limits.

Signal data limits have to do with the quality of input. In audio terms, we would describe such cases as having a poor ``signal-to-noise'' ratio. One example would be trying to hold a conversation with a jack-hammer running right beside you. Clearly, unless something is done about the ambient noise, you are never going to get beyond a certain level of communication, regardless of how hard you listen. Similarly, if text is presented on a CRT with a terrible font and a lot of glare on the screen, your reading speed will be limited despite your best efforts to the contrary.

Memory data limits are less intuitive, but equally important. In an earlier section we discussed how perception involved active processing. Memory data-limits have to do with this processing. Consider the preceding example of trying to read poorly presented text on a CRT. One of the processes at play in this situation is your use of world knowledge, your knowledge about what an ``A'' is, for example. Part of what affects your effectiveness at reading in this situation is your ability to use this kind of ``previous experience.'' It is reasonable that someone using this terminal for the first time will not be able to read a fast as someone who has experience. This introduces the notion that the paradigms used, or the base of stored past experience, can differ in quality. Furthermore, the quality of the stored paradigm influences the quality of the data available for processing. The quality of these stored paradigms is what is referred to by ``memory data-limits.''

The relationship between resource and data limitation in the performance of a sample task is shown in Figure 17. Once a minimum amount of resources have been allocated (point R_{min}), performance improves as resources are added, up to the point R_{dl}. To this point, the task is resource limited. Beyond R_{dl}, the task is data limited. Unless the data is improved, performance beyond this point will not improve, despite any additional resources that may be allocated.

In the sample task illustrated in Figure 17, the quality of data is the constant, and the resources allocated are increased. However, if the quality of the data were improved, the same performance would likely be achievable using fewer resources (or better performance achieved utilizing the same resources.) There is a trade-off between resources and data quality. In virtually all cases a primary design objective should *be to minimize resource consumption by improving the quality of the data*. Things like improved display quality and graphics design are techniques for pushing back signal data-limits. Training and the use of appropriate mental models are means to push back memory data limits.

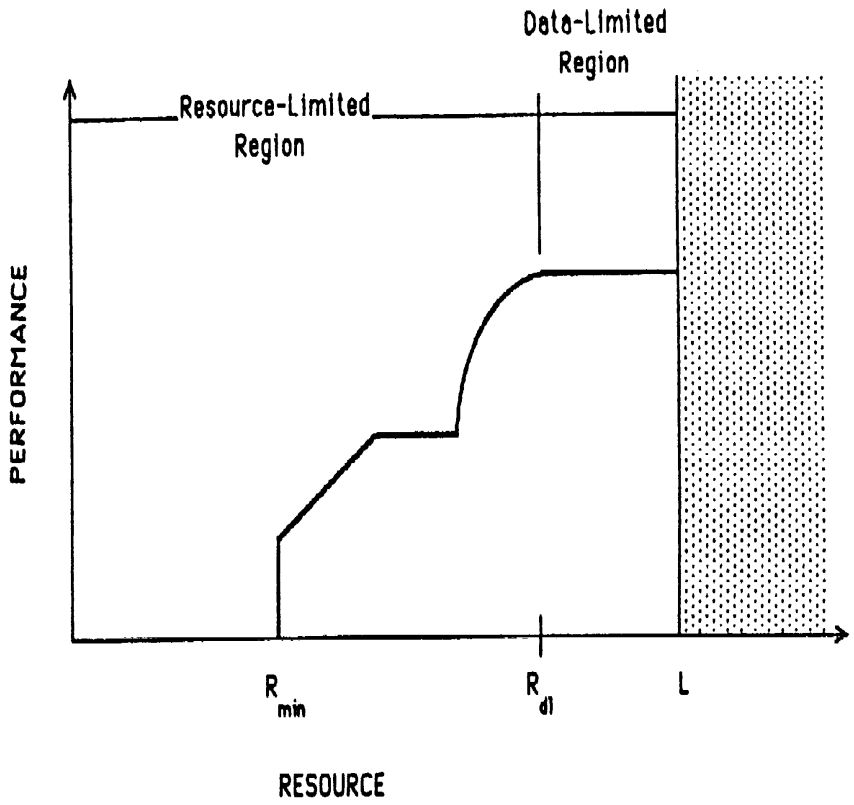

Figure 17: Data and Resource Limitation

Resource and data limitation in performing a simple task. Increase in performance is shown on the vertical axis. Increase in resources is shown along the horizontal axis. Certain minimal resources (R_{min})are required to initiate the task. Performance levels off at a certain point, where the task becomes data limited (R_{dl}). (from Norman and Bobrow, 1975)

Cognitive Load

Why make such a strong statement about minimizing the cognitive resources required to perform a task or complete a given quantum of work? Cognitive resources are in short supply and there is a high demand. Hence, there is a cost associated with their use.

A good measure of the complexity, or difficulty of a task is the amount of resources that it consumes (Moray, 1977; Kramer, Wickens and Donchin, 1983; Berlyne, 1960; Sheridan, 1980; Welford, 1978; Hartman, 1961). This is known as *cognitive load*. The concept of cognitive load is important, since load correlates directly with:

- learning time
- fatigue
- stress
- proneness to error, and
- inability to ``timeshare''.

Load is an important consideration in the design of both isolated tasks and large systems. At the individual task level, for example, one could consider the different load imposed in making a selection with a mouse having one, two, or three buttons respectively. (This is an example that has been studied experimentally. See Bewley, Roberts, Schroit & Verplank, 1983, for example.) By testing individual cases, it may turn out that the one-button case has the lowest load, since there is no overhead in determining which button to select. However, in the larger context, it may turn out that there is a penalty ``down the road'' which results from having only a one-button mouse. Consequently, rather than having reduced the overall load of the system, we may have just redistributed when or where the load is manifest.

We leave it as an exercise for the reader to debate whether this is the case with the one-button mouse on the Apple Macintosh computer *versus* the multi-button mice found on many other systems. For our purposes, suffice it to say that the example illustrates how complexity and design must always be considered in context, since almost everything has side-effects.

Interference

Given that resources are limited, sometimes a critical resource cannot be allocated. This may occur when performing a single task with a particularly high cognitive load. More often, it occurs when simultaneous demands are being made on the same resource by two (or more) different tasks. A simple example of this is the everyday experience of a parent having two children talk to them at once.

There are some situations, however, where we can do more than one thing at a time. This can occur when there are enough resources to go around. Listening to a radio program while driving would be one example. But what happens in an extreme case, such as when the car goes into a skid? Most likely the driver will stop listening to the radio, and concentrate on getting the vehicle back under control. Full attention will be focussed on the driving task, and no surplus resources will be available to concentrate on the radio (no matter how interesting the program may be).

Degradation in the performance of one task due to with another is known as *interference*. In some cases, processes will interfere with each other, resulting in a degradation in the performance of each. As with the driving example, however, what seems to happen more often is that the individual invokes some priority mechanism that determines which of the competing tasks is more important. It then allocates the resources under contention accordingly. Thus, the performance of the priority task continues at the expense of the other.

Since many of the tasks that are performed using a computer have a high cognitive load, they are susceptible to interference. One of the goals of the designer is to take steps to reduce the likelihood of this occurring. Reducing the load associated with task performance is one approach to accomplishing this. As discussed earlier, another way is to improve the quality of data available to the user.

Yet another way to approach reducing interference is to take steps to minimize the likelihood of competition. Different sensory modalities utilize different resources (Wickens, Sandry & Vidulich, 1983). Therefore, they are less susceptible to interference than those using a common one. For example, while tracking a moving target on a display (using a mouse, for example), you are more likely to be able to perform a simultaneous verbal task than another visual/manual one. An example of where this information might be applied is in the design of help mechanisms. We could base a design on the hypothesis that if we present help messages using the audio channel they will not interfere with an application which is presented *via* the visual channel. This may or may not be an improvement, but it is an example of how an understanding of the underlying cognitive structures can help suggest design ideas.

Problem Solving

At this point, there is a strong temptation to delve into a long discussion about the nature of problem solving, and all of the current theories that relate to it. That is what most writers do. But it is an involved topic, we do not have the space, and others (such as Lindsay & Norman, 1977) have already done an excellent job. For our purposes, we will restrict ourselves to making a few key points.

First, solving problems requires one's attention. It exhibits what is known as *attentive* behaviour. Second, problem solving consumes a relatively large number of resources. Consequently, problem solving and its accompanying attentive behaviour is highly susceptible to interference.

In working with a computer system, there are two classes of problem that confront the user: *operational* and *functional*. Operational problems have to with the *means* of performing work. Functional problems have to do with the *content* of that work. Imagine you were composing music with an interactive editor for musical notation. ``How do I delete that note?'', is an operational problem. ``Should I orchestrate this note with a flute or saxophone?'', is a functional one.

One objective of user interface design should be to minimize operational problem solving. All resources consumed at this level are being diverted from the primary application for which the computer was adopted in the first place. That is, *they are wasted*, insofar as performing the primary task is concerned. Design features such as consistency and careful documentation are critical in reducing this diversion of resources.

The overhead of functional problem solving can also be greatly reduced by careful design. The key here is recognize the influence of representation on the relative difficulty of solving a problem. This goes back to the old notion of ``representation as a tool of thought,'' or ``a problem properly represented is 3/4 solved.''

If a computer is adopted to help perform a particular task, chances are the task already taxed a human's problem solving ability. This being the case, it is critical that the system be designed so that the user can get to the heart of the problem by the most effective path, dissipating a minimum of valuable cognitive resources along the way.

Cognitive Skills

We have all encountered people who seemed to perform tasks effortlessly that we found exceedingly difficult. For us, if it could be done at all, the task clearly involved attentive, problem solving behaviour. Playing piano, sailing a boat, writing computer programs, or solving math problems are all possible examples. While experts in such activities are likewise performing complex tasks, their behaviour exhibits very different properties than our own. What characterizes their behaviour is that they are *skilled* in that particular activity.

A useful and important point made by Card, Moran and Newell (1983) is their contrasting problem solving *vs* skilled behaviour as two extremes along a continuum. Similarly, Rasmussen (1983) describes three levels of task performance: *knowledge-, rule-,* and *skill*-based performance.

Unlike the attentive nature of problem solving, skilled task performance is *automatic*. Skilled task performance consumes negligible cognitive resources, compared to problem solving. Consequently, skilled performance is less susceptible to interference. Also, the resources thus released can sometimes be allocated to some other task that may be performed synchronously with the skilled task. One visible clue that a subject is exhibiting skill is their executing two or more complex tasks in parallel (but the extent to which this happens is very task dependent).

Many of the properties of skilled and unskilled (i.e., problem solving) behaviours can be observed in a simple experiment in a Chinese restaurant. Have a meal with someone who has never used chopsticks, someone who has used chopsticks since childhood, and a couple people who lie in

between. For the novice, eating will consume their full attention. The slightest distraction will cause them to drop their food. A more interesting case is the subject who is slightly beyond beginner. While they may seem to eat effortlessly, you will see that if you ask them a difficult enough question (i.e., one that generates enough interference) they too will drop their food. With the expert, it is unlikely that any task that you present them that doesn't interfere with their hand or mouth will interfere with their eating. They can talk, read, recite poetry, or answer calculus questions without introducing problems. They are skilled.

One way to gain some insights about the nature of skill is to observe people who are experts at a particular task. This is precisely what is done by Card, Moran and Newell (1980) in the reading describing their study of computer text editing. What they discovered was that experts had built up a repertoire of techniques, or *methods*, for dealing with the standard situations encountered in editing a document by computer. One can think of these methods as the cognitive equivalent to macros, or subroutines. They bind together all of the individual steps, or *operators*, required to perform a particular learned task. In text editing, the expert need only recognize the problem, *select* the most appropriate method for dealing with it, and invoke it.

Because experts are highly practiced, and recognize each situation as one of something that they have seen before, the recognition and selection tasks impose minimal loading, and the execution of the task is automatic.

The metaphor of a cognitive subroutine is a fairly functional description of what is going on in skilled performance. Like executing a subroutine, the expert considers performing the skilled task as a single thing. In contrast, the novice has to ``write'' and ``debug'' the solution to each text-editing task on a step-by-step basis. Neves and Anderson (1981) have described the acquisition of cognitive skills as the process of *compilation* and *proceduralization* of these individual steps (what Card, Moran and Newell call *unit tasks*) into debugged, higher-level structures.

Skill Acquisition

Achieving a skilled level of proficiency in any task is difficult. It seems that skills - cognitive and motor/sensory - are learned through practice and repetition, pure and simple. This is seen in a classic study by Crossman (1959), who studied the acquisition of skills in cigar making. It is also covered in Newell and Rosenbloom (1981), who discuss a ``power law of practice''. Obviously, not all skills have the same difficulty of acquisition. Most of us can master riding a bicycle. Few, even if we had the opportunity, could learn to fly a helicopter or become a virtuoso on violin. The *learning curve* for a number of diverse skills is shown in Figure 18.

The study of the acquisition of cognitive skills is an important and developing field. The interested reader is referred, in particular, to Anderson (1981) and to Schneider (1985). The reader is also referred to the somewhat larger literature on sensory/motor skills, since much of it applies equally to cognitive skill. Welford (1976) is a good place to start.

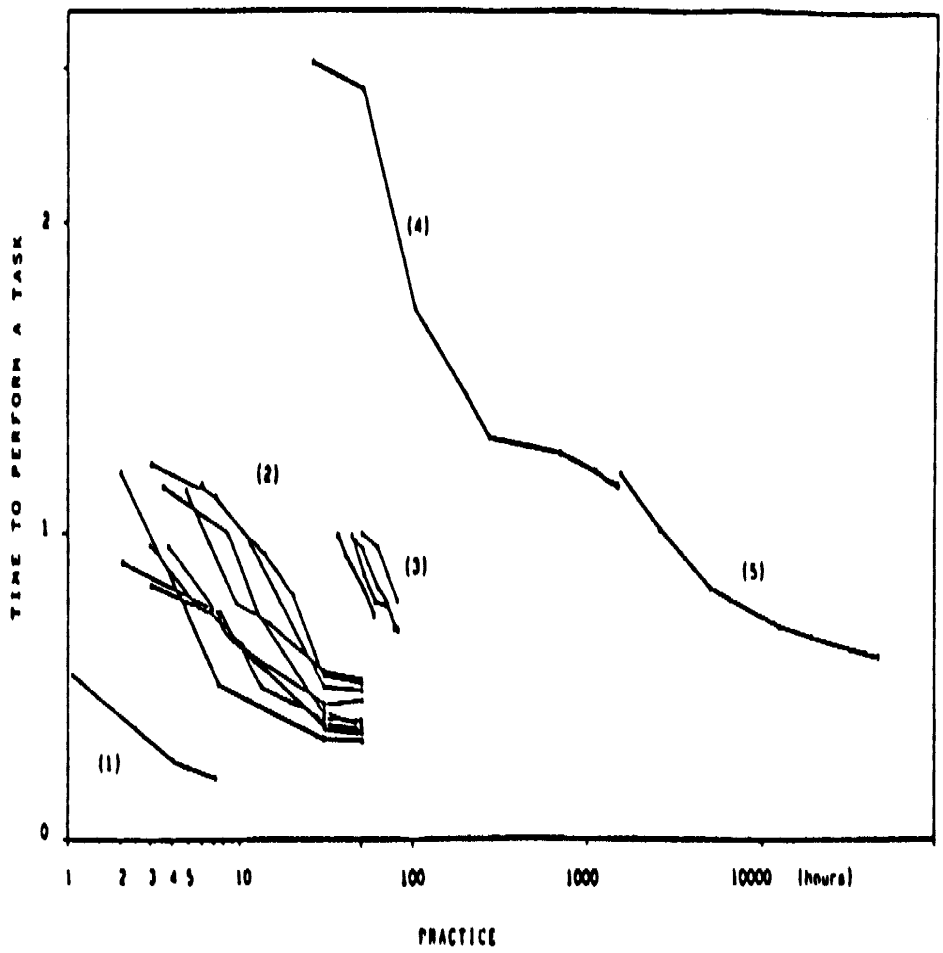

Figure 18: Learning Curves

Learning curves for a number of tasks are presented (Bosser & Melchior, 1985). (1) Reading inverted text (minutes/text): Average 6 subjects (Kolers, 1976). (2) Text processing with Wordstar editor (minutes/CET); Benchmark test with 50 core editing tasks (CET; 9 subjects). (3) Typing alphanumeric code material (seconds/keystroke); Average of 18 employees each group (Braddeley and Longman, 1978). (4) Mail sorting (seconds/letter); Average of 30 employees. (5) Cigar-making (minutes/cigar); Average of several employees (Crossman, 1959).

Training procedures are part of the user interface, and their design should encourage the development of skills in an isolated, controlled, and non-threatening way. For example, at Xerox Palo Alto Research Center, a program was developed to acclimatize users to using a mouse. It was a simple pursuit-tracking task with a twist: the target being tracked was represented as a fly, and the tracking symbol was a fly-swatter. The instructions given to the user were to kill the moving fly by getting the swatter over top of it and clicking the mouse button. As a result of this training, users were able to proceed to other tasks without any overhead being consumed by the operational problems of the mouse. Similar types of training were used by Buxton and Myers (1985) to rapidly train novices to simultaneously perform a continuous task with each hand.

The general rule here is that many skills developed in isolation transfer well to situations where the skill is used in conjunction with the performance of other tasks. There is also some evidence

that the skill is acquired more rapidly in this isolated situation (Schneider, 1985). As with all rules, however, this one does not always hold. For individual cases, field testing is the final proving ground.

Skill Transfer and Consistency

We now encounter a bit of a dilemma. Skilled performance has the desirable property of minimal loading during task execution. On the other hand, skill acquisition has been shown to be expensive. Is this not just a case of transferring the loading from execution time to learning time? Is there a net improvement, especially in the current climate that suggests that systems should be easy to learn as well as to use?

We can sidestep some of these questions by being clearer about what we are trying to accomplish. Let us stop talking about vague concepts like ``ease of use'' and ``user friendly.'' A more productive formulation of what we are trying to do is *accelerate the process whereby novices begin to perform like experts.* Implicit in this is a recognition that the qualitative difference between novice and expert performance is the exhibition of skilled behaviour.

While skills *are* difficult to acquire, new users don't come to a system completely vacant in this regard. Day-to-day life has equipped us with a large repertoire of highly developed skills, and the more specialized we are in any aspect of our work or pleasure, the more specialized those skills are likely to be.

The skills required to operate a system need not be learned from scratch. Generally, the more specialized the application, the easier it is to design around existing skills. For example, accountants can touch-type on numerical keypads and draftsmen can work a drafting machine with one hand while using a pencil in the other. These are both skills that can form the basis for a powerful *and appropriate* design. Analyzing the target population with respect to possible exploitable skills gives a whole new direction and bite to the often-heard platitude, ``know the user.''

In some cases, however, a new skill may be required. We may be able to get a head-start in training by using an existing skill as a point of departure. The use of metaphor is one aspect of this. However, metaphors can go beyond the use of icons, and can also include control functions, such as motor skills.

The use of existing skills as the basis for performing new tasks is known as *skill transfer*. There are four main criteria for successfully designing an interface to maximize this transfer:

- build upon the users' existing set of skills
- keep the set of skills required by the system to a minimum
- use the same skill wherever possible in similar circumstances
- use feedback to effectively reinforce similar contexts and distinguish ones that are dissimilar.

The idea here is that by keeping the repertoire of skills small, the skills get used a lot. This is consistent with the law of practice, in terms of the novice attaining a skilled level of proficiency.

Using the same skill in similar circumstances is a critical part of this. Besides the issue of practice, the underlying cognitive principle here is that if interfaces are *consistent* in what skills are used in a particular context, then the exploitation of in-system skills will be maximized. That is, in a well designed system, when the user is confronted with a new situation, all of the feedback mechanisms will shout out ``this is like this other thing which you have done before,'' and the user will be able to infer what to do by transferring what has already been learned to the new situation.

As a means of confronting the issues of consistency and transfer, we suggest that the reader work through the following exercise. Analyze the direct manipulation systems at your disposal, and consider the degree of consistency that exists among dragging, selection from pop-up menus, and drawing with rubber-band lines. Are these the same basic generic transaction? How

so? Can and should they be performed using the same motor skills? In what way is loading reduced if they are?

Crossman (1959) is an important early paper on skills acquisition. By its emphasis on methods and their selection, it laid much of the ground-work for the GOMS model of Card, Moran and Newell (1980, 1983). Payne and Green (1983) and Green and Payne (1984) are a source of discussion of issues pertaining to interface design, learnability and consistency. Polson, Muncher and Engelbeck (1986) present a model of transfer, and therefore a theoretical definition of consistency. In the paper, they make quantitative predictions of transfer effects, and test them experimentally. The paper is interesting for its approach as well as its results. Polson and Kieras (1985) is a related paper that discusses learning and performance in text editing. A study that is based on this work and which investigates transfer between different text editors is that by Karat, Boyes, Weisgerber and Schafer (1986). Transfer between two different tasks, text editing and graphics editing is discussed in Zieger, Hoppe and Fahnrich (1986).

Transaction Cost

One of the key challenges in marketing is determining the right *price point* for a product. The reality is, there is a fine art in determining the precise point at which a product should be priced, and that price point is generally very sensitive. Very small differences can make the difference between success and failure. Psychology is part of it[1]. Why are articles priced at $2.99 instead of $3.00? Because market research has demonstrated that there is a psychological barrier at play, and that 1 cent difference – which in objective terms is trivial - could make the difference between a product being purchased or not. But there is more to it than that. If the price point is too low, the product may sell extremely well, but the vendor will lose money, or not make enough to reinvest into the R&D to develop the next product – something essential to sustaining the company. On the other hand, if the price is too high, even if some people will still buy the product, again, the overall revenue (as opposed to that for a single sale) will be too low to sustain the business.

What this illustrates is that consumer behaviour is partially driven by some cost-benefit analysis, and that there are subtle and important thresholds in this equation.

I have made this sudden leap into the world of marketing because I believe that there is a very similar kind of “price point” when it comes to access to functionality in user interface design. In design, the domain and currency are different, but the underlying concepts are rather similar. In our case, instead of dollars, the currency in which cost is measured is cognitive load, learning time, and consequence of errors. The underlying take-away is that there is a cost to the user for every transaction, and thinking about transactions in terms of their underlying economics, using a cost-benefit model, for example, is extremely important and useful.

As in “purchasing” anything, transaction cost will affect behaviour.

> *Imagine that you are in Toronto and telephoning someone in Paris, but you get a busy signal. Your subsequent behaviour, as to when and how often you call back, will very much be governed by what kind of telephone you have. Do you have a rotary dial phone, a touch-tone phone, a phone with a redial function, or, a phone with an auto-redial key, which redials the last number until it gets through?*

What is important in this example is that we can assume that the caller knows *how* to redial in all cases and that there is nothing *other than the cost of the transaction* preventing them from doing so. And yet, the cost *will* affect how often the caller redials.[2]

---

[1] See for example, Bennett, Brennan & Kearns (2003) and Anderson & Simester (2003).

[2] Obviously, cost will not be the only factor. Behaviour will also be affected by the motivation for the call. If the call is urgent, they will redial more often, regardless of cost. If it is a casual call and the caller has other

This notion of *transaction cost* was introduced in 1937 by the Nobel Prize laureate Ronald Coase in his essay "The Nature of the Firm." This concept, which he applied to the economic foundation of firms, translates equally well to developing an understanding of the economics of interaction.

Appreciating and being able to articulate the importance of transaction cost and the cost-benefit analysis of user interfaces can help us address one of the most common hurdles that occurs in practical development situations, what we call the *you can already do that* myth. In practice, suggestions for implementing a particular work-flow or capability are met with the comment, “but you can already do that,” followed by some arcane description of how to do so. Since more often than not, this comes from the most senior engineer on the project, the suggestion is dropped, and the team goes on to address some other issue, such as adding a new feature.

Figure 19: The Controls on my TV.

Note that the TV comes with built-in controls for changing the mode (video or TV), volume and channel. So why do I need a remote control? It just provides the same functionality, and adds extra cost. To repeat a too-often heard phrase, “You can already do that”

What an understanding of transaction cost tells us is that any claim that “you can already do that” without a discussion of cost *is meaningless.* One of the most important skills of the user interface or usability expert on the team is the ability to intervene in such discussions and articulate the counter-argument in terms that are acceptable, appropriate, understandable and convincing

The phone dialing example is one way to make the point. The television example that follows is another.

> *Think about the remote control for your television, which I suspect is very much like mine, which is shown in Figure 19. Every essential control on it is also on the TV itself. Thus, the functions on the remote are redundant – you can already do that. Yet the remote control brings real value – despite this redundancy. That value lies in the reduction of the transaction cost in accessing those functions. In short, you don’t have to get out of your chair or risk spilling your popcorn when you go to change the channel, or turn the volume down. To move from this analytical perspective to a deeper experiential one, do without your remote for a week and see how that affects the experience of watching TV.*

---

things to do, the call may be postponed indefinitely. All of this is consistent with the cost-benefit model that we are suggesting is at play in the interaction.

> *Note that the impact of reducing the cost of changing channels went well beyond saving time or effort. Changing the transaction cost of the user pushing a button changed the very essence of television itself. Once viewers could change channels instantly, both commercials and programs had far more pressure on them to maintain viewer interest. The consequence of not doing so was losing the viewer to another channel. Television was changed forever – "simply" by adding redundant buttons that cost less to push.*
>
> *To one degree or another, the same potential exists in almost all products and services. The alert designer needs to factor these considerations into any decision that they make, and must be able to draw on examples such as these in order to communicate the importance of transaction cost to their colleagues.*

The phone dialing and television remote control examples help explain the basic concept of transaction cost, and how it can be used to counter the all too frequent "you can already do that" issue. However, its relevance is broader than that. Within the design process, it should be a consideration in comparing alternatives. "Will this work?" is an important question, but not sufficient insofar as making an appropriate evaluation is concerned. One must always add the question, "At what cost?" Furthermore, the answer to this question must not just be driven by an assessment of the cost of implementation, but as well, by the resulting transaction cost to the user. Taking both into account is critical to performing the cost-benefit analysis appropriate for evaluating different alternatives. And, whenever considering matters of cost, it is important to keep in mind that one of the more significant opportunities for creativity is around the currencies according to which one chooses to measure such questions of value.

In light of this cost-benefit economic way of viewing things, it is a simple step to apply what we know from the everyday world to the design of products and services, namely, even if desirable, some things are just too expensive. Yes, I would love a Porsche, yes I know how to drive it, and yes, I would enjoy doing so. So usability, desirability and emotional considerations are not the issue. The point, rather, is that if it costs more than I am willing or able to pay (in some relevant currency), then what does it matter?

This leads to an insight into what I see as one of the biggest – and most easily fixed – problems with the "feature check-list" mentality of the past. Briefly stated:

> *Specifying a feature without attaching an associated maximum transaction cost, in the appropriate currencies, will lead to a checked off list, not a well-designed product.*

Let's just assume that the people who bring new products to market are well intentioned and want to excel at what they do. If a product or service is specified as a list of features, what better way to improve the product than to add features to that list? Such an attitude is as human as it is wrong – something attested to by the feature-bloat of too many of today's products. If, on the other hand, there is a cost associated with both features and higher-order transactions, the desire to exceed the specification will be redirected towards lowering the cost rather than adding a feature. The underlying concept here is this:

> *The specification of a product or service is an explicit reflection of our values. The more clearly this is articulated, the more the specification will drive behavior in the desired direction.*

Finally, if the transaction cost lies above what we might call the user's *threshold of frustration*, then the underlying functionality *does not exist* for that user in any meaningful way. This is true regardless of documentation, training programs, and what others can demonstrate. As our earlier discussion of cognitive load, interference, etc. make clear, despite individual differences and our ability to learn, human capability is finite. So are the resources that any individual can allocate to any given task at any given time. Design that is not sensitive to this is analogous to someone in marketing setting the price of a product without any thought to what the market will bear, or what is the optimal price point. This would be unacceptable in marketing. So should it be in systems design. Stated another way:

> *Until our sensitivity to transaction costs matches the subtlety and fine degree of granularity seen in the understanding of price points in product marketing, HCI must still consider itself an immature discipline.*
>
> *The take-away from this discussion is this: the specification of a product is not done until there is a maximum and minimum transaction cost assigned to each feature and capability.*
>
> *Finally, remember, the bias of the path of least resistance is one of the most important biases on human behavior. Least resistance equates very strongly to least cost – at least in the sense that we are speaking about it here. If you want to encourage a particular behavior, cost control is your primary tool. But in so doing, don't fall into the trap of assuming that obtaining minimum cost for everything is the goal. Typically, even if that were possible, it is frequently not what you want to do. The simplest example of this can be found in video games. If the goal was to make applications as easy as possible, then the best video game would have one button, and as soon as you push it, you win.*
>
> *Yes, this is an extreme example, On the other hand, good game designers are perhaps the best reflection of the what can be attained by those who know how to manage this world of transaction cost.*

While such considerations have always been important, we have managed to more-or-less get away without taking them fully into account. I believe that such days are over. The market today is no longer impressed by that fact that something can be done, or if it kind of works. Standards have changed as the industry matures. But perhaps equally or more important, the larger ecosystem in which things function is – as part-and-parcel of that maturity – becoming far more complex. In making decisions about the design of a browser, for example, what might have worked just fine on a desk-top or lap-top computer may be a disaster on a touch-operated mobile device. As developers, given the realities of the cost of decisions in terms of the full life-cycle of a product or service, we cannot generally afford to engineer anew our technologies for every new product that emerges. Hence, pragmatics dictates that we have to perform careful analysis right from the start, before implementing anything, that takes this larger ecosystem into account. In my opinion, a deep understanding and consideration of transaction cost is a critical component of any such analysis.

Compatibility

When the cause-and-effect behaviour encountered in working with a system matches the user's expectations, it is said to have good *stimulus-response (S-R) compatibility*. Having good spatial congruence between items in a menu and the layout of function keys used for making selections is a good example of S-R compatibility. Since the pairing between menu items and function keys is clear, we can expect that training time and operating load will be reduced.

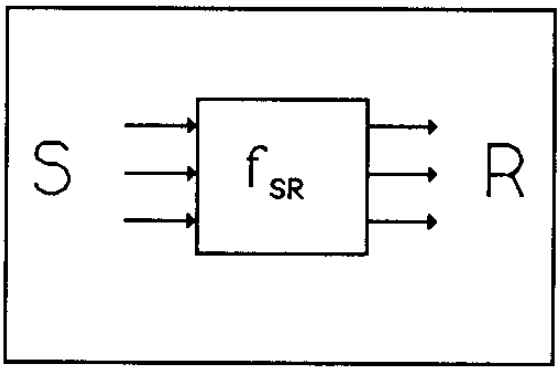

Figure 20: Compatibility as a Mapping Function
We can consider S-R compatibility in terms of a function that maps stimuli onto responses. Compatibility varies with the inverse of the complexity of the mapping function.

A good way to think of compatibility is in terms of a transfer function, f_{sr}, that maps stimuli onto responses (Figure 20). Essentially, compatibility varies inversely with the complexity of this mapping function.

There are two main forces that drive compatibility: *spatial congruence*, as was seen in the menu-function key example, and *custom*. That much of compatibility *is* custom, and is learned, can be seen from the example of a toggle light switch. In North America, the rule is generally ``up is on, down is off.'' However, in many other countries, such as England, the convention is just the opposite. There are many comparable learned conventions, or *stereotypes*, that may be used in the design of a user interface. The meaning of colours or words are examples. Like the light-switch, these are usually culturally or professionally dependent. Their meanings can vary widely, and the designer must be aware of potential problems.

Some examples can help us develop a better understanding of other aspects of compatibility and stereotypes.

Suppose that we are running an acoustics simulation and we want to increase the frequency of a waveform. The controls that we might use are shown in Figure 21. For each, what direction would you move the control to increase the frequency? Try to formulate a concise rule that states how to increase and decrease the parameter being controlled using each type of controller.

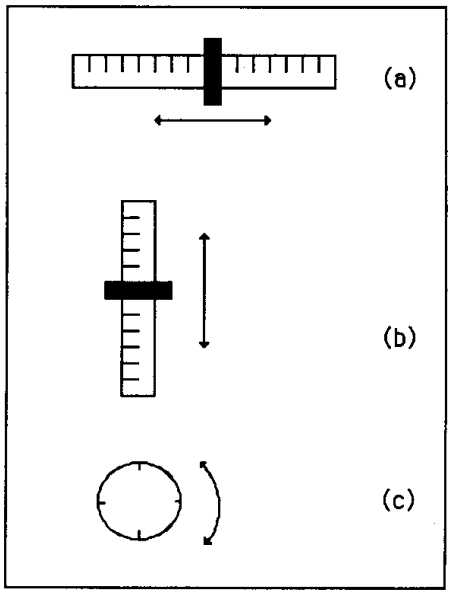

Figure 21: Three Potentiometers

For each type, indicate which direction increases the parameter being controlled. Formulate a general rule for each case.

There will probably be a fairly high consensus that ``increase'' corresponds to a right, up, and clockwise motion, respectively. However, what increases must be consistent with the user's expectations. For example, if the user is interested in the waveform's *period*, rather than its frequency, then the control's effect on the waveform should be reversed (period being the inverse of frequency). Subtle differences can have a strong affect on compatibility. Understanding the user's mental model is critical.

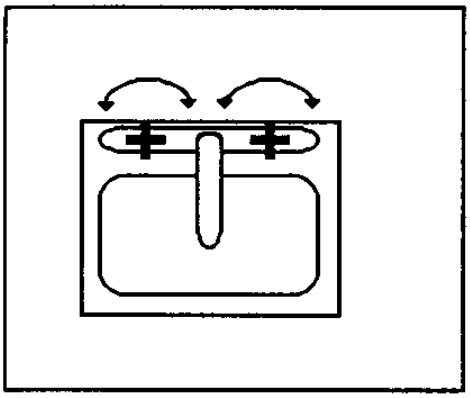

Figure 22: Water Faucets

Which faucet is for hot water and which is for cold? Indicate what direction each faucet is turned to turn the water on. (from Smith, 1981)

Water faucets introduce another concept about compatibility. Look at the faucets illustrated in Figure 22. Which faucet is for hot water and which is for cold? What direction should each faucet be turned in order to turn on the water? Try to generalize the rules that you use in answering each of these questions.

In North America, there will be fairly strong consensus that the hot water tap is on the left. However, there will likely be some variance as to which direction to turn the tap to get the water

flowing, especially with the hot water tap. The degree of consensus is a good indicator of the strength of the S-R stereotype.

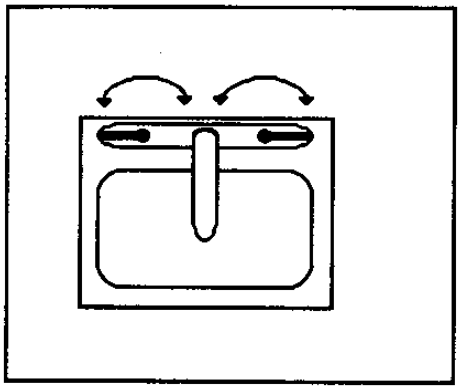

Figure 23: Lever-Type Faucets

If the faucets shown in Figure 22 are replaced by lever-type faucets, how are your expectations affected? Is the direction that you would turn the taps to turn on the water affected? (From Smith, 1981).

What happens if the faucets of the previous example are replaced by lever-type faucets, as illustrated in Figure 23? Are our expectations the same with respect to how to turn the water on? From the previous example, most North Americans would have formulated the rule: ``Faucets are closed by clockwise motion and are opened by counterclockwise motion.'' We now have to qualify this rule. The supplement is: ``If the faucet has lever-type controls, it is opened by pulling and closed by pushing.''

Seemingly simple changes can affect our expectations and behaviour. As with changing the tap handles in the previous example, minor changes in graphical presentation or input devices can have a strong effect on user's expectations about system behaviour. As a designer you must understand these effects so that they work for you, rather than against you.

The next example illustrates this type of error in an actual system. On the original Apple Macintosh there was a rotary potentiometer to control the screen brightness. Its position with respect to the computer is illustrated in Figure 24.

Before reading further, look at the figure, then close your eyes and mentally turn up the intensity of the display. Which way did you turn the potentiometer?

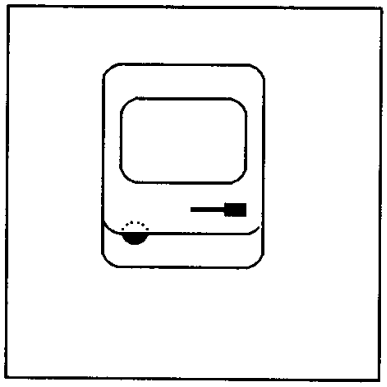

Figure 24: The Intensity Control on the Original Macintosh Computer

A rotary potentiometer was used to control intensity. Which way should the pot turn to increase the intensity of the screen?

Since a rotary potentiometer was used, the designers automatically assumed that the ``rotate clockwise to increase'' rule applied. While well intentioned, this wiring conflicts with our expectations and is wrong. Why? The problem arises because, only the bottom half of the potentiometer is exposed. Therefore, the rule that the user applies is one that we saw with the horizontal linear potentiometer: ``move right to increase, left to decrease.''

For further information on compatibility, the reader is encouraged to read Smith (1981). This paper has several good examples, many of which have been used in this chapter. Fitts and Seeger (1953) and Fitts and Deininger (1954) are important early papers. Barnard, Hammond, *et al.*. (1981) discuss compatibility with respect to the ordering of elements in command languages. Finally, an interesting attempt to refine the theory of compatibility can be found in John, Rosenbloom and Newell (1983) and Rosenbloom (1985).

Summary

The use of computing systems makes heavy demands on our cognitive system. Performing many tasks imposes a high cognitive load. This can be partially dealt with by adopting training procedures that develop skills appropriate for the application. This, however, imposes its own cost in terms of the time and the effort that must be invested before expert performance can be achieved.

Taking a broader perspective, we see that every system can be characterized by the skills that are required to utilize its full functionality. This we will call the *prescriptive model* (PM) (Bosser, personal communication). Similarly, we can characterize users by the set of skills with which they approach the system. This we will call the *descriptive model* (DM).

Typically, the descriptive model is a sub-set of the prescriptive model. These two concepts provide a means to illustrate the relationship between the two main components of *cognitive engineering*: training and design.

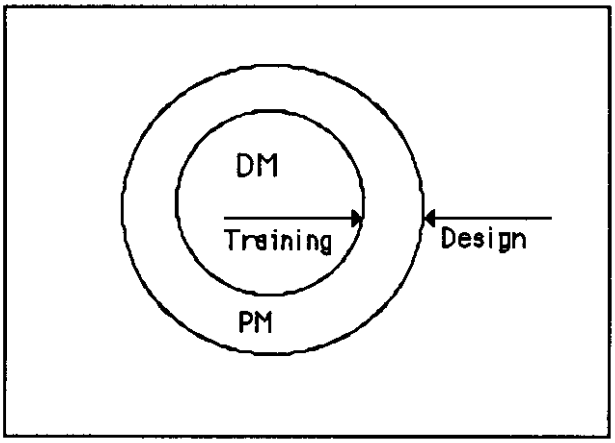

Figure 25: Training and Design

The gap between system demands and user capabilities are shown by the difference between the prescriptive model (PM) of the system and the descriptive model (DM) of the user. In the ideal system, PM=DM. The objective of training is to push the PM outwards. The objective of design is to push the DM inwards.

In Figure 25 we see that the role of design is to minimize the difference between the PM and the DM. In this context, training is seen to be a necessary evil which compensates for the shortcomings of the design.

Good design will take serious account of the cognitive issues discussed in this chapter. It will minimize the operational hurdles that users must overcome in order to achieve their primary task. Where there are shortcomings in design, cognitive principles must again be employed to develop effective training procedures. While not providing an exact science, the cognitive principles discussed will hopefully help suggest good design alternatives. The theory should provide a starting point. But testing and evaluation must always serve as the final proving ground.

Chunking and Phrasing

Introduction

It is no secret that the user interface of most computer systems could be improved. Systems are often intimidating, prone to error, and require a high investment in effort before productive work can be undertaken. A desire to make systems easier to use is a good starting point, but we can't get very far without a theory of how to do so.

"Easier to use" is easy to say, but it suggests little about *how* to reduce errors and frustration and promote faster learning. In order to make some headway in this direction, we might best reformulate the problem as "How can we accelerate the process whereby novices begin to perform like experts?". Underlying this formulation is an assumption that there is a qualitative difference between how experts and novices achieve particular goals. This assumption is supported by much of the literature in problem solving and the acquisition of cognitive skills (e.g., Fitts, 1964 & Anderson, 1980).

Experts and novices differ in the coarseness of granularity with which they view the constituent elements of a particular problem or task. Novices are *attentive* to low-level details. For example, *operational* details such as finding a particular character on the keyboard or remembering the name of a command involve *problem solving*. The result is that valuable cognitive resources are diverted from the central problem at hand.

With experts[1], these low-level details can be performed *automatically*. Hence, the size of the chunks of the problem to which they are attentive are much larger. The *skills* that permit these tasks to be performed automatically, however, must be highly learned, usually through repetition (Newell & Rosenbloom, 1980). The acquisition of skills, therefore, can be characterized by developing an ability to perform ever-larger chunks of a problem automatically.

Somewhere in between these two extremes lies what Rassmussen (1983; 1986) calls *rule-based behaviour.* This is behaviour which – while appearing to be skilled, still requires attentive action. Fitts (1964) likewise characterizes the three stages of skill acquisition as*, cognitive, associative*, and *autonomous*.

We can now return to our reformulation of the problem at hand, "How can we accelerate the process whereby novices begin to perform like experts?". Our premise is that there should be as close a match as possible between the structure of how we think about problems or goals, and the language or representation that we use in solving or achieving them. In what follows we argue that this can be achieved by engineering the *pragmatics* of the human-computer dialogue (Buxton, 1983) to reinforce the chunking that we believe would used by an expert working in the domain. Another way of stating this is that the dialogue structure, especially the pragmatics, can be engineered so as to maximize *compatibility* (Fitts & Seeger, 1953; John, Rosenbloom & Newell, 1985) with the task.

[1] At the risk of being redundant, it must be emphasized that while what is being said applies to novice/expert performance in general, for the purposes of this discussion, we are talking about *operational* expertise; that is, expertise in operating the system, not expertise in the relevant task domain. So, for example, while the operator may be an expert animator, they might be a novice in computer usage.

Syntax: Two Approaches

The design of the syntax has a major effect on the quality of the user interface of an interactive system. It affects learnability, the frequency and nature of user errors, the retention of skills (as with non-regular users) and the speed of task performance. A major problem for users is the cognitive load imposed by remembering the tokens of a command and their order (see, for example, Barnard, Hammond, Mortan, Long & Clark, 1981).

One approach that designers have taken to avoid such problems is to limit the number of arguments to a command. The user interface of the Macintosh computer, for example, limits operators to having only one explicit argument. This causes problems, however, for operations such as *move* which require both a direct and indirect object. To get around this, applications such as *MacWrite* (Apple, 1984) replace the single command *move* with two lower-level commands *cut* and *paste*. While the new primitives have a simpler syntax, the user's mental model must be restructured to map the concept move onto these two new primitives. Rather than simplifying the user interface, therefore, it is possible that the single-operand-per-verb strategy simply redistributes the cognitive loading.

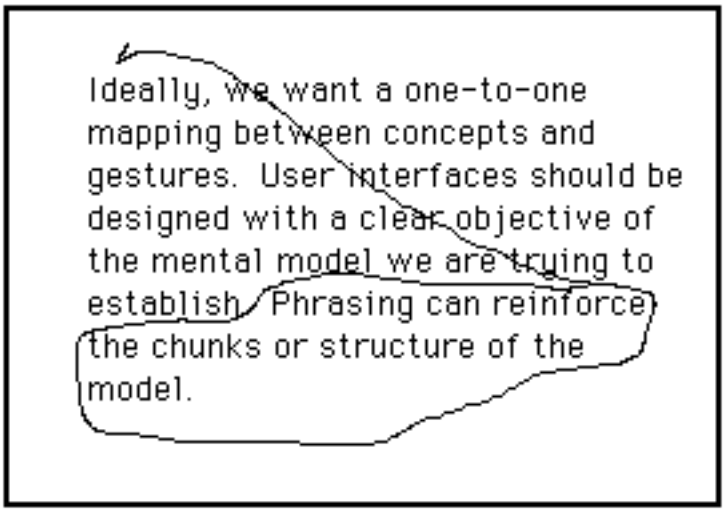

Figure 26: Proof-Reader's Symbol Specifying "Move."
Contrast the directness of this with the "cut-and-paste" strategy utilized by MacWrite (Apple, 1984).

An alternative design strategy exists. If *move*, for example, is the primitive that most closely corresponds to the user's model, then the design problem is to use it while minimizing the burden of remembering the arguments and their ordering. Proof-reader's symbols offer one approach to doing so. An example is shown in Figure 26.

There are at least three points worth noting about this example, especially in contrast with the "cut-and-paste" strategy for specifying the same operation:

- the entire transaction, verb, direct object, and indirect object are all specified in a single gesture;
- there will never be an error in syntax since the ordering is implicit in the gesture;
- the operation is specified using existing skills and does not require restructuring of existing mental models.

Phrasing and Gesture

We can think about the components of the move command in the previous example as woven together by a thread of continuity similar to that that binds together a musical phrase. The

"statement" is initiated in a state of neutrality. It is articulated by a continuous gesture, and upon *closure*, it returns to neutral state where another phrase can be introduced by either party. As in music, the phrase is characterized by tension (in this case muscular) and the neutral state delimiting the start and finish by relaxation.

One of our main arguments is that we can use tension and closure to develop a phrase structure to our human-computer dialogues which reinforces the chunking that we are trying to establish. [Tie in Kendon(1980), and (1986), especially pages 34-35 w.r.t. *Gesture Phrase*, *Idea Units*, and related work.

In the "body-language" of haptic input, kinesthetics and muscular tension are the raw materials of establishing a phrase structure. The underlying basis for this is an extension of *the Yerkes-Dodson Law*, illustrated in Figure 27 (Yerkes & Dodson, 1908). This law asserts that as "arousal" increases, that performance follows an inverted *U*: increasing to a certain point, then falling off as arousal passes some critical point[1]. What we argue is that kinesthetics and muscular tension serve a similar purpose in stimulating arousal during task performance. Hence, with the gesture comes heightened arousal and performance and in the periods of relaxation, there is an implicit, but clear indication that it is alright to be interrupted, or move on to the next step.

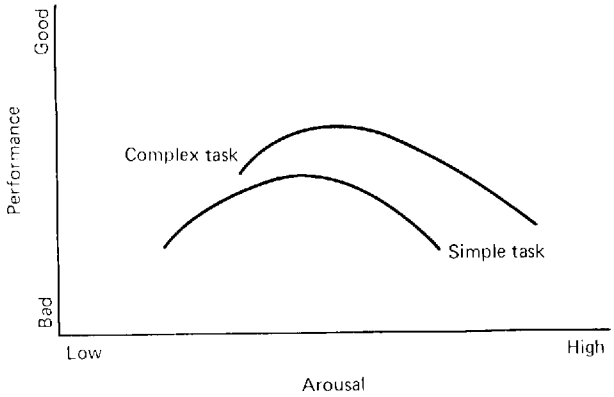

Figure 27: Yerkes-Dodson law relating performance to arousal (From Kantowitz & Sorkin, 1983, p. 606)

Compound Tasks

Problems that we saw previously in the syntax of a single command also appear at another level of the human-computer dialogue. In actual applications, many of the transactions which we perform consist of *compound tasks*. Selecting an electrical component and positioning it in a circuit board layout would be an example of a *selection/positioning* task (Buxton, 1982). Similarly, identifying a word by finding it in a document and then selecting it would be an example of a *navigation/selection* task (Buxton & Myers, 1986). In many such cases, we would argue that the user models the compound task as a single entity. In such cases, having to address the sub-tasks independently may result in an additional burden comparable to using *cut* and *paste* instead of *move*. Furthermore, we claim that phrasing through kinesthetic gesture can overcome this problem.

Pop-up menus provide a good example to illustrate our point. In general, one would consider making a selection from a pop-up menu as a single task. However, on closer examination, it consists of three sub-tasks:

[1] For a simplistic example, at the leftmost side of the x axis, consider you are sitting in an easy chair trying to read. Your performance will be low. Now, consider that you are in the middle of an exam hall, reading an exam paper. Now, your performance will be very high. Finally, moving to the extreme right, consider trying to read a procedures manual to determine what to do to in the middle of a nuclear power plant going critical. In this case, your reading performance will be nearly nil.

- *invoke the menu* : by depressing the mouse button;
- *navigate to selection* : by moving mouse while button is depressed;
- *make selection and return* : release mouse button.

In this case, the "glue" that ties the three sub-tasks together is the tension of holding the mouse button down throughout the transaction. By designing the dialogue in this way, errors of syntax and mode errors are virtually impossible to make since the concluding action (articulated by the mouse button being released) is the unique and natural consequence to the initial action (depressing the mouse button). Furthermore, the tension of the finger holding down the button gives constant feedback that we are in a temporary state, or *mode*. (There is a slight irony to this, since it is precisely in so-called "modeless" interfaces that pop-up menus are most commonly found.)

Phrasing and Cognitive Skill

In their 1983 study, Card, Moran and Newell discussed how experts collapsed low-level text editing tasks into cognitive "subroutines" that they termed "routine cognitive skills". Anderson (1982) describes the acquisition of such skills as based upon the compilation and proceduralization of knowledge about the underlying sub-tasks. We believe that phrasing can organize these sub-tasks to accelerate this process.

Pragmatics and the Components of Output

If Card, Moran and Newell's routine cognitive skills are compilations of lower-level primitives, one could try to determine the basic building blocks. One answer comes from the generic input tasks of Foley, Wallace and Chan (1984):

- *Select* an item in 1, 2, or 3D;
- *Position* an item in 1, 2, or 3D;
- *Orient* (rotate) an item in 1, 2, or 3D;
- *Path* : specify a path, such as in inking in a paint program;
- *Quantify* : specify a numerical value;
- *Text* : enter text, as in word processing

On closer examination, however, we see that these primitives are not necessarily all at the same level. Let us use the *position* primitive as an example.

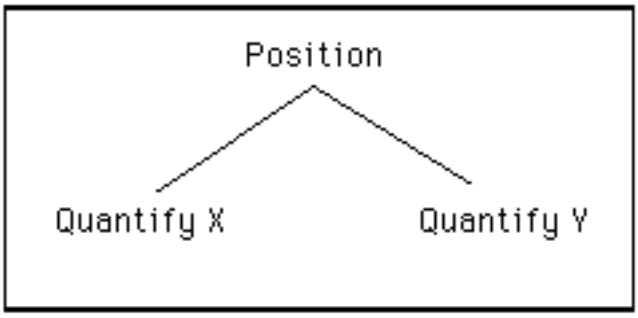

Figure 28: Position as an Aggregate of 2 Quantify Tasks

If we use a mouse or a tablet, positioning an object in 2D becomes a single task. However, the moment that we change transducers and use a QWERTY keyboard, specifying the same coordinates involves two primitives, namely *quantify X* and *quantify Y,* as illustrated in Figure 28.

We see from this example that even Foley, Wallace and Chan's six primitives have a deep structure. Whether the sub-tasks are consciously perceived, however, is very much influenced by the gesture (and capturing transducer) used. When appropriate, a single gesture (pointing) can articulate a single concept (position).

We can build further upon the previous example. Let us look at a simple system for transcribing common music notation (Buxton, Sniderman, Reeves, Patel & Baecker, 1979). Notes are entered using a simple short-hand notation. Using a stylus and digitizing tablet, the user points at where a note is to appear and enters one of the shorthand symbols shown in Figure 29.

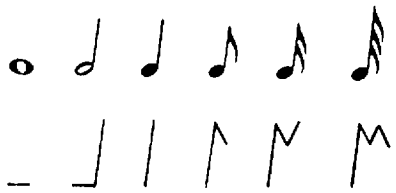

Figure 29: Short-Hand Symbols for Transcribing Musical Notation.
(From Buxton, Sniderman, Reeves, Patel & Baecker, 1979.)

Using this system to enter a 16th note is shown in Figure 30.

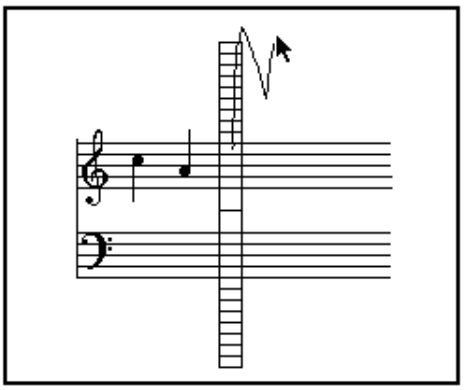

Figure 30: Entering a 16th Note Using a Single Gesture.

The underlying structure of adding notes using this technique is shown in Figure 31. We see that adding a note, like positioning, is actually made up of a number of sub-tasks. However, when implemented as described, these sub-tasks all collapse into the single primitive *add note*.

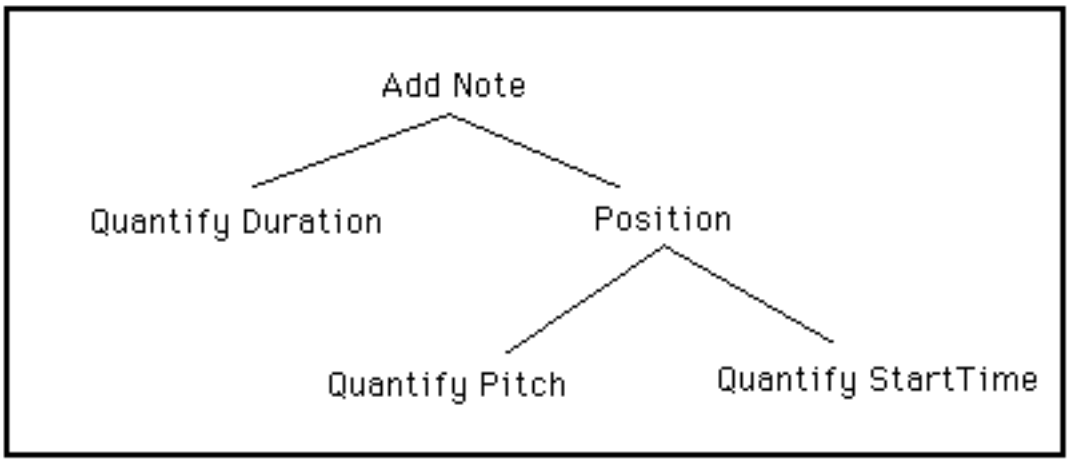

Figure 31: Task Hierarchy in Add Note Task

Grammars, Pragmatics and Simplicity

If the sub-tasks of a higher-level concept like "Add Note" can collapse into a single gesture, is there any model that helps predict the savings? One approach, based on recent studies, attempts to obtain a measure of the difficulty of a user interface by analyzing its underlying grammar. This work was pioneered by Reisner (1981), and further developed by Green & Payne (1984) and Green, Payne, Gilmore and Mepham (1984).

In her original work, Reisner developed a set of heuristics which she used to analyze the grammar of the interaction language of a particular system. From this analysis she would derive a value which gave a measure of the system's learnability and proneness to error. The heuristics were based upon:

- Number of productions
- Number of terminals
- Length of productions

We can apply such an analysis to the grammar of our Add Note primitive, shown below (non-terminals start with upper case, terminals start with lower case):

AddNote:= quantifyDuration PositionPitchTime

PositionPitchTime := quantifyPitch quantifyStartTime

We can apply an approximation of Reisner's heuristics on this grammar in which we assume that the weight of each production and each terminal is unit1. Since we have two productions and three terminals, the total weight is order 5.

However, if we use the character recognition technique described above, we would argue (from experience) that the *real* weight of the entire transaction is closer to the weight contributed by a single terminal, namely weight 1. Our explanation for this is that the user need not be attentive to any of the operational details of the component sub-tasks. The complete concept can be expressed in a single fluid compatible gesture.

Gesture and Pragmatics

The notion of physical gesture is central to virtually all of the examples discussed. In each case, the key to using a particular gesture rests in the appropriate transducer. Conversely, the main limiting factor, restricting the range of available gestures is the sorry state of current practice in

input. The lack of pressure sensitive devices (such as mouse buttons to control line thickness), foot controls, and two-handed input are just a few obvious examples.
To this point all of our examples have involved sequential bindings. However, as changing gears with a manual transmission illustrates, the binding among related tasks can be in parallel and across limbs. This is demonstrated in Buxton & Myers (1986).

Conclusions

We have argued for user interface pragmatics to accelerate the acquisition of expert operational skills. The key is gesture-based phrasing to chunk the dialogue into units meaningful to the application. Any concept or transaction that can be described in a single word or phrase should be able to be articulated by a single gesture. This one-to-one correspondence between concept and gesture leads towards interfaces which are more *compatible* with the user's model.

The work described is based on practice and experience rather than formal experimentation. It is preliminary, and a great deal of research remains to be done. However, the examples discussed are sufficiently persuasive to warrant an examination of current design practice.

Memory, Motor Action, and Skill

Introduction

This section ties together some of what we know about human memory and aspects of some of the concepts discussed in this chapter – including skill, cognition and motor-action. In so doing, we introduce the notions of *pragmatic* and *epistemic* action (Kirsh & Maglio, 1994).

A very preliminary sketch of contents to be included here is summarized in the following tentative points **(I do mean preliminary and tentative – these are quick notes serving as a place-holder for me while writing and need proper checking and references added before I stand by what is written**):

Two aspects of memory:

1. Short term memory and chunking
    - Give overview to Miller (1956).
2. Recognition *vs* Recall
    - people are generally better at recognizing things than recalling them
    - *test-edit-test* provides a tactic that enables one to exploit our superior recognition strategy
    - the generally multiple *test-edit-test* is often still better than pure recall
    - while *in-head* visualization is likely used by some for some memory tasks, *external* visualization, by motor actions in the physical world appear to be frequently used instead
    - external motor-action assisted visualization to support recognition may be the primary method, compared to in-head, when possible – although it must be hard to quantify this, since in-head visualization for the purpose of recognition is hard, if not near impossible, to observe, quantify or qualify, in contrast with external visualization

Moving from memory tasks to cognition in supporting new situations, such as reported in Kirsh & Maglio's, 1994 paper on playing Tetris. Hence (and here I am likely making huge unjustified/unsupported leaps):

In terms of cognition:

- Much of the cognition involved in dealing with problems in new situations consists of recognizing familiar patterns and/or components of the task, and applying what we have learned in the previous case to the new situation

- A good example is in games like chess. While the expert player may never have been confronted with the exact situation or board layout encountered at a particular time, the expert will see patterns in the configuration of pieces on the board that are familiar, with which they are experienced, and therefore they can draw on that experience in informing the strategy behind their next move.
- The ability to recognize such patterns enables the player to see things at a higher level (deep structure rather than surface structure) and this "chunking" is a fundamental aspect of cognitive skill.
- The key lesson here is that even when confronting new situations, recognition is a fundamental component of the task, as well as of cognitive skill.
- This observation provides a link between cognitive skill and what we know (and discuss above), about memory.
- In the chess example, the player has the ability to, and often does, move a piece, without committing to the move, in order to (presumably) aid in seeing such patterns.
- Due to the pragmatic constraints in terms of what a player can actually physically manipulate on the board at any given time, the ability of a player to do in-head visualizations, especially in terms of visualizing several moves ahead, is a fundamental aspect of skilled performance.
- In this, it is important to keep in mind that it would be a mistake to assume that such visualizations are literal snapshots of entire board configurations, rather than subsets, or higher order representations based on the patterns or relationships among the relevant meaningful "chunks". That is, the recognition mechanism can work with other encodings than just literal representations of the physical world.
- Presumably physically moving individual pieces tentatively (without committing to the move) as an aid to recognize the consequences of a particular move is far more common with non-expert players, who – due to a lower level of skill – are less adept at in-head recognition of patterns.
- This would be easy to observe, and therefore easy to determine. On the other hand, this assumption may be wrong, since it contradicts the findings in Maglio & Kirsh (1996), which found that incidences of epistemic action *increased* with skill, rather than decreased. It would be interesting to test the chess case. It could either confirm Maglio & Kirsh's findings, or show that the result is task dependent (and therefore may lead to a refinement of the underlying theory).
- Some tasks are harder to visualize in-head. Different geometric transformations provide a good example: it is easier to recognize many 2D shapes as the same when their position is translated in the X and Y domain, than when there is no translation but they are rotated around the Z axis.

- The cost of the extra motor actions required to manipulate the world in order to support external recognition (i.e., epistemic action) is more than made up in the resulting improved performance compared to a reliance on in-head recognition. (Maglio, Wenger & Copeland, 2003; 2008).

Paul Maglio comments:

(1) We did not look at this kind of thing in chess, and I agree that expert chess players probably would not see much benefit from taking physical actions for their epistemic effects. And I agree too with your basic reasoning here relating to perceptual chunks in chess. For expert chess players, the patterns are over-learned, precompiled. In my dissertation I argued that Tetris was not like chess in this way -- there were simply too many Tetris patterns to precompile and it so it was not cost effective to do. Later Eric Demaine actually proved Tetris is NP-Hard (whereas chess is not), see http://arxiv.org/abs/cs/0210020 . And I actually tried to make an argument along these lines in my dissertation, but it was more in terms of cognitive plausibility than in terms of computational complexity (formally).

(2) So I think that for certain tasks and in certain contexts, sure, the benefits may not

outweigh the costs. The details matter -- what is the specific computation that is being carried out and how can it be effectively distributed across the internal and external environments to simplify or speed up processing? In some cases, it might be useful and some cases it might not be.

(3) The only other study I did related to this was with the game Scrabble:

Maglio, P. P., Matlock, T., Raphaely, D., Chernicky, B., & Kirsh D. (1999). Interactive skill in Scrabble. In Proceedings of Twenty-first Annual Conference of the Cognitive Science Society. Mahwah, NJ: Lawrence Erlbaum, pp. 326-330.

where we found that "using your hands helps" you generate more words from a set of letters -- *when finding words is difficult*. That is, if there are lots of words hidden in the anagrams of a set of letters, letting folks physically rearrange them did not help, but if there were fewer words hidden in there, using hands did help (relative to not using your hands).

This is of course consistent with the story I am spinning here.

(4) So, finally, yes, you are asking great questions. The point is that details matter, both of the external environment and the internal processes, and their possible coordination. The main point of the epistemic action stuff is that we need to take external actions into account sometimes to understand people's effective strategies -- in the old days, this kind of internal-external explanation was simply not cool.

(2) Additional notes from BB: Need to add section on Hick-Hyman Law. Good approach is via ref: Seow, S.M. (2005). Information Theoretic Models of HCI: A Comparison of the Hick-Hyman Law and Fitts' Law. Human Computer Interaction, 20(3), 315-352.

Section on haptic / kinesthetic / proprioceptive … sensing, (i.e., motor-sensory) and cognitive aspects of input.

Quick notes on sources to consider:

- Gunnar's work: http://www.psyk.uu.se/forskning/forskningsprojekt/?languageId=1#tocjump_20775283083775686_19
- Camille's list: http://www.partly-cloudy.com/wiki/library:haptics
- Burdea, G.C. (1996).
- Boff, K.R.. Kaufman, & J. P. Thomas (Eds.)(1986), *Handbook of perception and human performance. Vol 1: Sensory processes and perception. Vol. 2: Cognitive processes and performance*. New York: Wiley. Vol 1: Chapters 12 & 13 / Vol 2: Chapter 31

Re Chunking – incorporate Kendon's Gesture Phrases. Kendon, A. (1986). Current Issues in the Study of Gesture. In Jean-Luc Nespoulous, Paul Perron & André Roch Lecours (Eds.). *The Biological Foundations of Gestures: Motor and Semiotic Aspects*. Hillsdale: Lawrence Erlbaum. 23-47.

•