# THE PSYCHOLOGY OF USER-COMPUTER INTERACTION

S. K. Card and A. Newell

**Rapporteurs:**

**Abstracts:**

## 1. User-Computer Interaction and Cognitive Psychology

Computer science is an asymmetric field. While many topic problems of computer technology are intimately connected with the human use of computing machinery, only the machine side of these problems is usually studied. An obvious topic that would benefit from work on both the human and machine aspects is human-computer interaction. However this is not easy to do. The need to base human factors for computer interaction on a theory of human cognition is discussed, stressing the primacy of design and the critical roles of engineering calculations. The ground is laid for such an advance by presenting the Model Human Processor, a highly approximative, but useful, engineering-level model of how humans proces information.

## 2. Examples of Model-based Human Factors

A series of examples are presented for how to model human-computer interactions in ways that permit approximative engineering calculations. The examples are drawn from pointing devices and editors. In each case, the advantages of having models or theories of the phenomena, as opposed to merely the results from comparative experiments are stressed.

## 3. Human-Computer Interaction in the Computer Science Curriculum

The point of view manifest in the prior two lectures provides a basis for discussing how human-computer interaction might become a part of the computer science curriculum. Nine issues raised by the introduction of human-computer interaction topics are discussed. Among other things, the conclusions are drawn that human engineering of actual interfaces will probably be done predominantly by computer scientists as opposed to human factors practitioners and that the teaching of human-computer interaction should be dispersed among subject courses rather than being concentrated into a course on humans. A suggested course syllabus is drawn to stimulate discussion.

LECTURE 1

# USER-COMPUTER INTERACTION AND COGNITIVE PSYCHOLOGY

In recent years, the convergence of a number of forces has brought the human-computer interface forward as an area of increasing importance within computer science:

1. There is finally enough raw computing power to make it practical to expend significant resources on the interface.

2. Bitmapped raster displays have reduced the cost of graphics interfaces to where they can become widespread and made graphics easier to program.

3. The effect of the above developments has been to make it possible and practical for computers to be applied to new tasks, tasks that tend to be more cognitive and that tend to require more intimate interfaces with higher densities of userinteractions/time.

There now seem to be nearly unlimited possibilities for interacting with computing machines far beyond the teletype-oriented turn-taking of the recent past. At the same time, interfaces have become both the critical element on which the success of many programs depend and the part of the program least predictable for designing.

Our goal in these lectures is to consider this subfield of human-computer interaction and its place in the discipline and teaching of computer science. The construction of computer interfaces is a topic of many parts: user interface management systems, control structures, data structures, graphical interaction techniques, to name a few. These are topics to which the computer scientist is naturally drawn. We intend, therefore, to focus on a part of the problem that, although acknowledged important, is something of a nuisance in human-computer interaction, namely, the human.

Computer science, as it currently exists, is a curiously asymmetric field. Many problems in computers are intimately connected with the human use of computing machinery. Yet only the machine side of this interaction tends to be studied. For example, programming languages mediate between, on the one hand, something that generates a program, and on the other, something that receives a program and executes it. One would expect there to be substantial technical study on each side of the problem: On the machine side there might be studies of parsing algorithms, algorithmic efficiency, run-time environments, and storage. On the human side there might be studies of what languages are easy to learn, to understand, to generate, and to debug. A

similar analysis could be repeated for other topics where computers and humans interact: the design of computer-based communications systems for organizations, team organization for software production, programming, human-computer interaction.

Of course, it is all well and good to puff about how computer science ought to be a bit less asymmetric, paying attention to the user, whereas the actual doing of this is a matter of some difficulty. The question is whether studies on the human side are possible with any acceptable degree of rigor. As a means of answering this question, we shall proceed with a particular view of the state of the art in human-computer interaction that fits what we think is needed for its progress and integration into computer science. We begin with a consideration of what is known about the human from psychology and human factors.

## Classical Human Factors

There have always been applied parts of psychology: intelligence testing, personnel selection, and clinical psychology, to state some examples. The study of how people deal with machines got its major impulse during World War II with studies of vigilance (why, after several hours on watch radar operators miss blips), aircraft cockpits (knobs and dials), and manual control (the extension of control theory to include explicit models of humans). Incidently, this latter study, manual control, was so successful that it is today becoming of limited importance: theories of pilot control are good enough that they can sometimes be used to replace the pilot in many of the same tasks that these theories used to be used in for predicting the pilot! Contemporary cognitive psychology is a direct descendent of these equipment studies (although, curiously, it has avoided manual control).

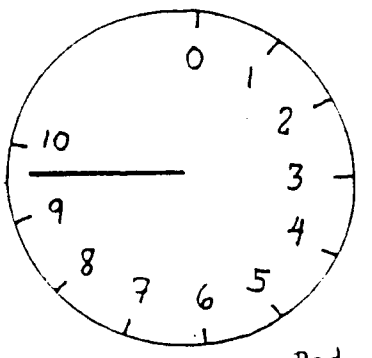
In the United States, studies of people and equipment came to be called human factors; in Europe, a similar (though definitely not identical) area of studies came to be called ergonomics. The classical style of human factors work has been the *comparative experiment.* For example, Fig. 1 shows a set of dials taken from an experiment on the check-reading of instruments. The users in the experiment (Kurke, 1956) are given a task to do. During this task the dial on the pointer is moving. The experimenter measures how long it takes the subject to notice that the pointer has gone into the "danger" zone. From Fig. 1, we can see that the dial displaying a red wedge whenever the pointer is in the danger zone takes the least time to notice and the dial with no marking of the danger zone is the worst.

For another example, Fig. 2 shows two possible arrangements of key pads for push-button telephones. One arrangement matches that of adding machines, the other arranges the keys in order. Users in this experiment (Conrad and Hull, 1968) "dialed" a set of telephone numbers with the result that more errors appear to have been induced by the adding machine arrangement than

HUMAN FACTORS EXPERIMENT 1:
CHECK-READING OF INSTRUMENT

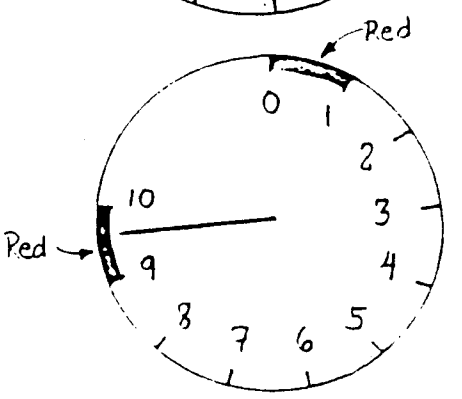
(a) No indication of "danger" zones (0–1; 9–10).

Time to notice = 27.8 s




(b) Red line indicating "danger" zones.

Time to notice = 20.5 s




(c) Red wedge appears when pointer is in "danger" zone.

Time to notice = 4.3 s

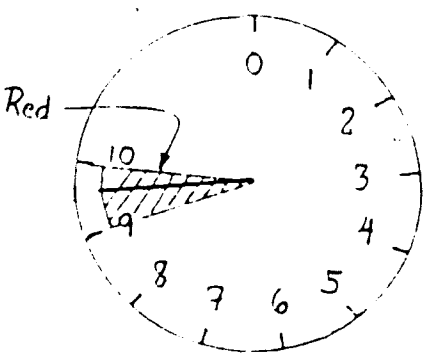



**Fig. 1.**

HUMAN FACTORS EXPERIMENT 2:
ERROR RATE OF TELEPHONE NUMBER PADS

| ADDING MACHINE | TELEPHONE |
|---|---|
| 9 8 7 | 1 2 3 |
| 6 5 4 | 4 5 6 |
| 3 2 1 | 7 8 9 |
| 0 | 0 |
| Percentage errors: | |
| 8.2 % | 6.4 % |

**Fig. 2.**

by the numerical arrangement. Partly on the basis of this experiment, telephone keypads in the United States are numerically arranged.

In addition to the comparative experiment, another tool of the human factors practitioner is what might be called *task analysis*, broadly construed: The analyst writes down everything a person must do to complete a task. He uses this list to discover conflicts, awkwardnesses, potential errors, and likely execution times. For example, the airline industry conducts "time-line analyses" in which each action a pilot will perform in, say, landing his airplane is carefully scrutinized.

Of course, there are other detailed methods available in human factors (see National Research Council, 1983), but these two, comparative experiments and task analysis, are really the essence of what is available. These two tools *are* useful in investigating human-computer interaction, but they have important limitations: The problems with comparative experiments are (1) they cannot be done at design time (because the system has not been built yet) and (2) with so many potential sources of interaction around, one does not trust them to generalize to the next situation (in the absence of some theory or model). The problem with task analysis is that, as usually practiced, the analysis has a difficult time reaching highly cognitive issues, and the systems we build are increasingly cognition-intensive.

## The Nature of Human-Machine Interaction

In order to gauge the purchase available from different analysis and measurement techniques, it is important to appreciate the full complexity of the human-computer interaction problem. The most general formulation of human-machine interaction is probably the case where a computer is embedded in a larger piece of equipment controlled by a human operator. Fig. 3 presents a diagram based on Sheridan's model of supervisory control (see National Research Council, 1983, Ch. 4) of a computer as part of some system such as a power plant or a space station. The computer is shown broken into two computers, one to service the task, the other to service the user interface. Evident in the diagram are a number of feedback paths indicating possible control loops for the sensors (e.g., automatic light adjustment), the activators (e.g., inner loop controls for aircraft), the display itself, and the controls the user employs. But complex interactions are also possible between the task interface computer and the human interface computer. Table 1 gives ten interaction modes between the user and the machine. A designer must choose amongst these very carefully. Of course many computing systems are simpler; but they can be thought of as degenerate cases of Fig. 3 in which the task, sensors, and effectors are all inside the computer and the human interface computer is just a user interface management system, or perhaps even code intermixed with task code.

# GENERALIZED MODEL OF HUMAN-COMPUTER INTERACTION

## (Based on Sheridan's Supervisory Control Model)

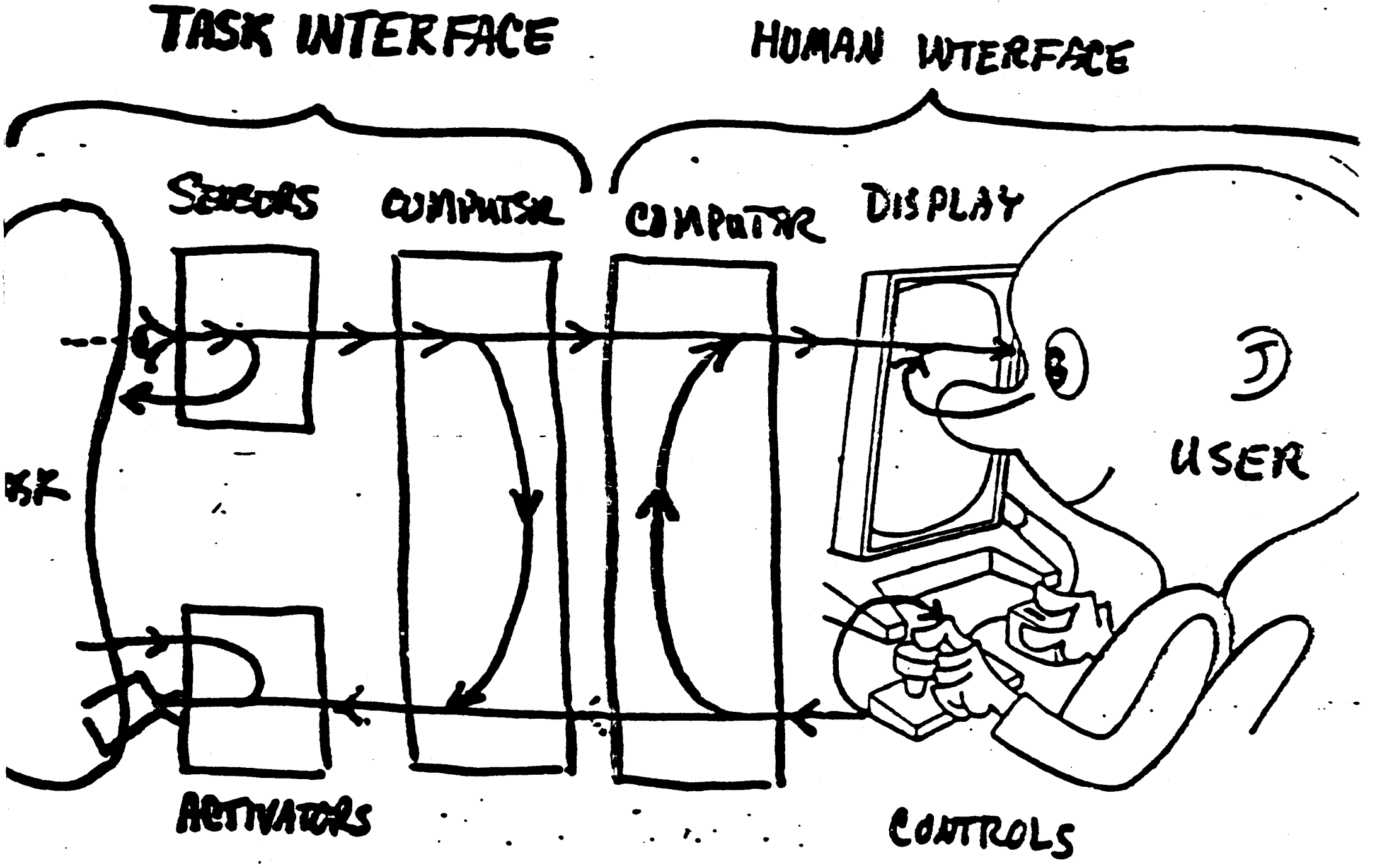



Fig. 3.

## TABLE 1.

### Levels of Automation

| | |
|---|---|
| **100% Human Control** | 1. Human considers decision alternatives, makes and implements a decision. |
| | 2. System suggests set of decision alternatives, human may ignore them in making and implementing decision. |
| | 3. System offers restricted set of decision alternatives, human decides on one of these and implements it. |
| | 4. System offers restricted set of decision alternatives, and suggests one, human may accept or reject, but decides on one and implements it. |
| | 5. System offers restricted set of decision alternatives and suggests one which it, the system, will implement if human approves. |
| | 6. System makes decision and necessarily informs human in time to stop its implementation. |
| | 7. System makes and implements decision, necessarily tells human after the fact what it did. |
| | 8. System makes and implements decision, tells human after the fact what it did only if human asks. |
| | 9. System makes and implements decision, tells human after the fact what it did only if it, the system, thinks he should be told. |
| **100% Automated Control** | 10. System makes and implements decision if it thinks it should, tells human after the fact if it thinks he should be told. |

Source: Sheridan, reprinted in (National Research Council, 1982, p. 112)

A number of problems must be resolved in designing an interface. Table 1 and the feedback loops in Fig. 3 have already illustrated the *automation problem* (which parts of the task should be done by which parts of the machine and who should have the initiative when?). Other problems are the *display problem* (how can the display be employed to increase system usability?), and the *interaction techniques problem* (which interaction techniques are advantageous?). Both of these are part of the larger *communication problem* (how is it that the intentions and information of the user can be communicated to the machine and vice versa?). Like the automation problem, the analysis of communication can become complex. There are logically three active agents involved, two computers and one human. Each of these may contain models of the others such that one agent may presume to do what it thinks another requires on the basis of its model of that agent, without actually being commanded. Intelligent tutoring programs, for example, choose which exercises to display to the user on the basis of a changing model of the user's competence (Sleeman and Brown, 1982). We shall not discuss these problems of interface design directly. Instead we shall discuss an approach to the problems and in so doing try to illustrate where we believe the study of human-computer interaction fits into other knowledge of computer science.

It is a basic goal of work on human-computer interaction to make it possible to have a discipline of interface design more like standard engineering. For example, it would be useful to be able to make back-of-the-envelope calculations of human performance. Why? Not just to emulate the physical sciences, but because design time is the critical time for human-engineering. And to analyze user behavior at design time we need predictive models. In fact, we can summarize the sort of models we need in terms of three criteria: task analysis, calculation, and approximation.

*Task analysis.* We need to be able to take a human task and to analyze what the rational courses of action are to determine the actions that will be required to accomplish the task.

*Calculation.* We need to be able to make at least simple calculations concerning things like time to complete a task, time to learn a system, and errors likely to occur. Furthermore, if calculations are to be useful they must only require data that is available and they must be reasonably simple.

*Approximation.* Humans, of course, are complex. If the calculations are to be simple (in fact, if they are to be done at all) they will have to be approximations. Fortunately, many of the decisions we have to make do not depend on the fine details of human performance.

The source of ideas for moving in the above direction is modern cognitive psychology. A major revolution in our understanding of human cognition has taken place since the 1950s. This increase in insight about human mental functioning ultimately derives from our new understanding of man as a processor of information. Advances in cognitive psychology, especially in the problem solving area, are closely related to developments in artificial intelligence, specifically with the development of techniques for representing complex information and with the theory of heuristic search. Many of the results from this area are highly compatible with the tools needed to study human-computer interaction (not so suprising since we are considering man in the context of information-processing machines).

## The Model Human Processor

At the center of any attempt to make use of cognitive psychology for engineering purposes is a summary of human information-processing capabilities. We shall, therefore, begin with our own such summary called the Model Human Processor. Here we shall only have time for a brief summary. The complete model is contained in Card, Moran, and Newell (1983). This model is inspired by the simplified description sometimes used in computer science to describe computer systems in terms of processors, memories, and switches (Siewiorek, Bell, and Newell, 1981); we shall describe the human processor in terms of processors and memories. In the spirit of approximation, the idea is to put forward a simplified engineering model for supporting predictions of user behavior, rather than to explain in detail the many intricate phenomena of human performance.

The Model Human Processor summarizes human processing capability in terms of three processors (see Fig. 4):

a Perceptual Processor,
a Cognitive Processor, and
a Motor Processor.

For single shot tasks, such as pressing a button at the sound of a signal, these processors run in a series. But for some tasks, the processors can operate pipelined-parallel: a user can type the last word while reading the next word, for example. (Actually there are several Perceptual Processors—speech and vision, for example, require different processing—but we have simplified this here.) These processors are thought of as having a processing cycle during which elementary processing takes place. As examples, events that occur within one Perceptual Processor Cycle tend to be perceived as part of a single event and movements are composed of simple micromovements.

# LONG TERM MEMORY

$\delta = \infty$ $\mu = \infty$ chunks

## WORKING MEMORY

VISUAL IMAGE STORE

$\delta = 200$ msec

$\mu = 17$ letters

AUDITORY IMAGE STORE

$\delta = 1500$ msec

$\mu = 5$ letters

$\delta = 7$ sec

$\mu = 3$ chunks

$\mu^* = 7$ chunks

PERCEPTUAL PROCESSOR

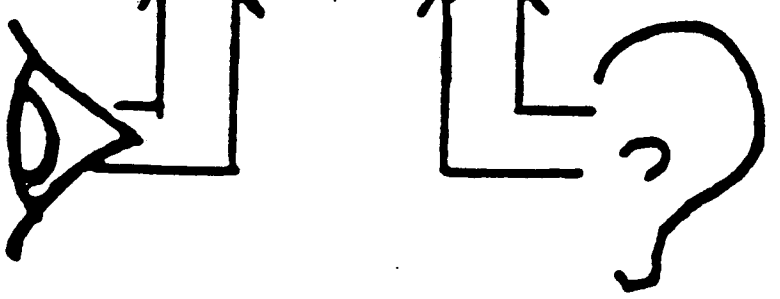
$\tau = 100$ msec

COGNITIVE PROCESSOR

$\tau = 70$ msec

MOTOR PROCESSOR

$\tau = 70$ msec



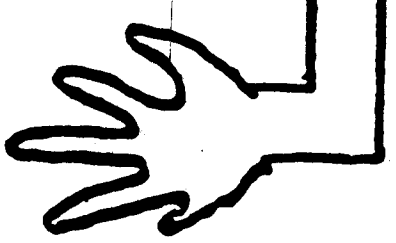


Fig 4

Human memory is summarized in terms of four memories:

a Visual Image Store
an Auditory Image Store
a Working Memory, and
a Long-Term Memory.

Fig. 4 does not look like the usual computer memory with separate transfer paths, because human memory is actually hierarchical. Items in the Visual Image Store and the Auditory Image Store appear after a short delay symbolically coded in the Working Memory. Working Memory can be thought of as a set of activated nodes of a semantic network in Long-Term Memory (where all the knowledge resides). The Visual Image Store and the Auditory Image Store are the sample-and-hold buffers where sensory inputs are stored in a physical code (if the light is brighter, the memory of it takes longer to decay). Working Memory is sort of the general register of the mind and stores information symbolically, generally with some sort of acoustic (or even visually-based) code. It is shown surrounding the sensory buffers because these are closely connected: About 10 msec (below the time grain of our model) after the physical pattern for a letter appears in a sensory buffer, it appears in Working Memory with a symbolic code.

Many of the limits of the human processor can be summarized in terms of a few parameters:

(for memories)

- $\delta$ Decay time
- $\mu$ Capacity, and
- $\kappa$ Code type

(for processors)

- $\tau$ cycle time

For example, if a long list of numbers is read to a person and he is unexpectedly asked in the middle to recall as many numbers back as possible, he will be able to recall about $\mu_{WM} = 3$ numbers. But, if a person is tested for his ability to recall 4-digit numbers, then 5-digit number, then 6-digit numbers, etc. He is likely to start having difficulty above $\mu_{WM}^* = 7$-digit numbers. In this latter case he is employing not only Working Memory, but also some Long-Term Memory as well to give the famous $7 \pm 2$ number.

Fig. 5 summarizes the values of this set of parameters. The values available in the literature for the parameters vary somewhat depending on the exact operational definition employed in an

## CONSTANTS FOR THE MODEL HUMAN PROCESSOR

| | | |
|---|---|---|
| **MEMORIES** | | |
| LTM | Decay | infinity |
| | Capactiy | infinite |
| | Code | semantic |
| WM | Decay | 7 [5 - 226] sec |
| | | 7 [5 - 34] sec (3 chunks) |
| | | 73 [73 - 226] sec (1 chunk) |
| | Capacity | 3 [2.5 - 4.1] chunks |
| | | 7 [5 - 9] chunks (with LTM) |
| | Code | acoustic, visual |
| VIS | Decay | 200 [70-1000] msec |
| | Capacity | 17 [7 - 17] letters |
| | Code | physical |
| AIS | Decay | 1500 [900 - 3500] msec |
| | Capacity | 5 [4.4 - 6.2] letters |
| | Code | physical |
| **PROCESSORS** | | |
| Perceptual | Cycle time | 100 [50 - 200] msecs |
| Cognitive | Cycle time | 70 [25 - 170] msecs |
| Motor | Cycle time | 70 [30 - 100] msecs |
| Eye movement | Fixation | 230 [70- 700] msec |

**Fig. 5.**

experiment, which experimental paradigm is used to measure it, or which human subjects are measured. This is the sort of variation that drives psychologists to distraction and despair. What is amazing is that there is, in fact, a good deal of concordance in the results of psychological experiments if only one spends as much time searching for the similarities as for the differences. For example, there must be hundreds of experiments that discover an operation with a time constant on the order of .1 sec. These regularities are generally underappreciated, probably because of the extreme emphasis in psychology on detecting statistically significant differences rather than in looking for approximations and idealized models. Many engineering disciplines based on the physical sciences also have to contend with variation: heating design (what temperatures will the building experience?), soil engineering (the composition of soils in a piece of land can only be sparsely sampled), bridge building (what will the wind speed and gusting behavior be? what will be the exact structural consequences of rust?) We handle this problem of parameter variation by having a lower and upper bound on the parameter set by the literature. Since these bounds represent unlikely extremes, we also set a more typical number. So we write the cycle time of the Perceptual Processor as

$$\tau_P = 100\,[50\sim200]\ \text{msec.}$$

The lower bound is 50 msecs; the upper bound is 200 msecs; and 100 msec is a typical value that appears under typical light levels in many experiments. This is the form in which parameters appear in Fig. 5.

## Deriving Parameters from the Psychological Literature

Evidence for the value of these parameters comes from experiments in the psychological literature. Fig. 6a shows measurements of the decay time for the Visual Image Store. An array of letters is shown to a subject for a very short interval (50 msec). At some interval after the letters disappear a visual or auditory cue indicates to the subject which row of letters he is to report (He cannot be expected to recall more than the effective capacity of Working Memory $\mu_{WM}$*, even if these letters were in the Visual Image Store, so a cueing arrangement is used to sample). The curve plots the extra letters (in excess of Working Memory capacity) that the subjects could report as a function of the time that elapsed before they were cued. The curve shows an exponential decay whose slope we can characterize by the half-life $\delta$. The family of curves shows that there is more regularity that we could parameterize (the number of letters visible has an important effect on the half-life). In this case we judge the gain is not worth the increase in fussiness for application, and

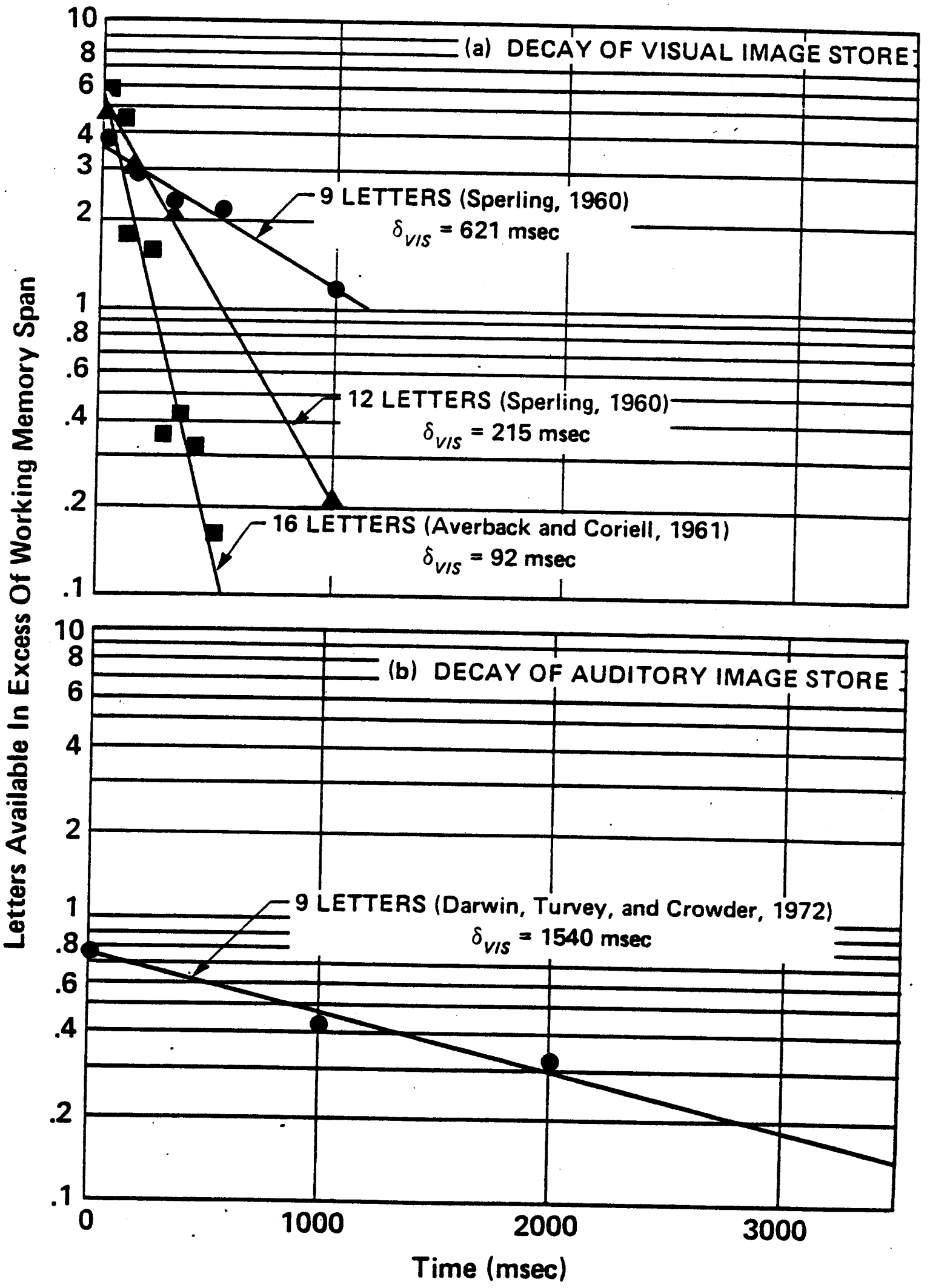

(a) DECAY OF VISUAL IMAGE STORE
9 LETTERS (Sperling, 1960)
$\delta_{VIS}$ = 621 msec
12 LETTERS (Sperling, 1960)
$\delta_{VIS}$ = 215 msec
16 LETTERS (Averback and Coriell, 1961)
$\delta_{VIS}$ = 92 msec
(b) DECAY OF AUDITORY IMAGE STORE
9 LETTERS (Darwin, Turvey, and Crowder, 1972)
$\delta_{VIS}$ = 1540 msec
Letters Available In Excess Of Working Memory Span
10
8
6
4
3
2
1
.8
.6
.4
.2
.1
0
1000
2000
3000
Time (msec)

**Fig. 6.**

so we accept the wider range of the parameter rather than a smaller range but more difficult application. Fig. 6b shows a similar measurement for the Auditory Image Store.

Fig. 7 illustrates that the unit of memory is a chunk (information treated as a single unit) and that the number of chunks makes more difference than what the chunks are composed of. The subject is given either one chunk or three chunks to remember, then maliciously prevented from rehearsing by having to count backward by sevens. After a time, he is asked to recall the original chunk. The curve plots the percentage of chunks correctly recalled as a function of time. Again there is roughly an exponential decay that can be characterized by a half-life.

As a final example of the origins of these parametric values, Fig. 8 gives estimates of the Cognitive Processor rate. One series of experiments involve presenting to the subjects a number of numbers or other items, then giving them a test item. The subjects are to say whether the test item was one of the original set. It has been found that this task takes time linear with the number of items in the original set. Currently accepted models of the task have the subject mentally matching the test item one at a time with members of the set, and hence it is one estimate of cognitive cycle time. Experiments with everything from numbers to random shapes give times in the neighborhood of 27 to 93 msec/item. Another way to perform the measurement is to show a person a number of dots or other shapes and see how long it takes for him to say how many there are. Depending on how many there are, the shape, and the person, this number ranges from 40 to 172 msec/item. The measurement with the slowest times is silent counting. A person is asked to count to some number (saying the unit digits silently to himself) and the time measured. This measurement gives a number of about 167 msec/digit. When we put these and other measurements together, we find that there is a range, but that they are all somewhere in the neighborhood of a tenth of a second. Our summary of the literature is $t_C = 70[25 \sim 170]$ msec.

## Principles of Operation

In addition to the architecture and parameters we have talked about, we also need to describe a sort of Principles of Operation to capture some of the dynamic behavior these miss. Fig. 9 lists the Principles of Operation for the Model Human Processor. A few comments about these:

*Recognize-Act Cycle.*

The human does not have a fetch-execute cycle like a classical computer (that is, a plan with a pointer). Instead, on each cycle, data that is in Working Memory matches best something that is in Long-Term Memory. The match becomes the new contents of Woking Memory. Another way of saying this is that certain nodes of Long-Term Memory are activated. On each cycle these activate other nodes, and so on.

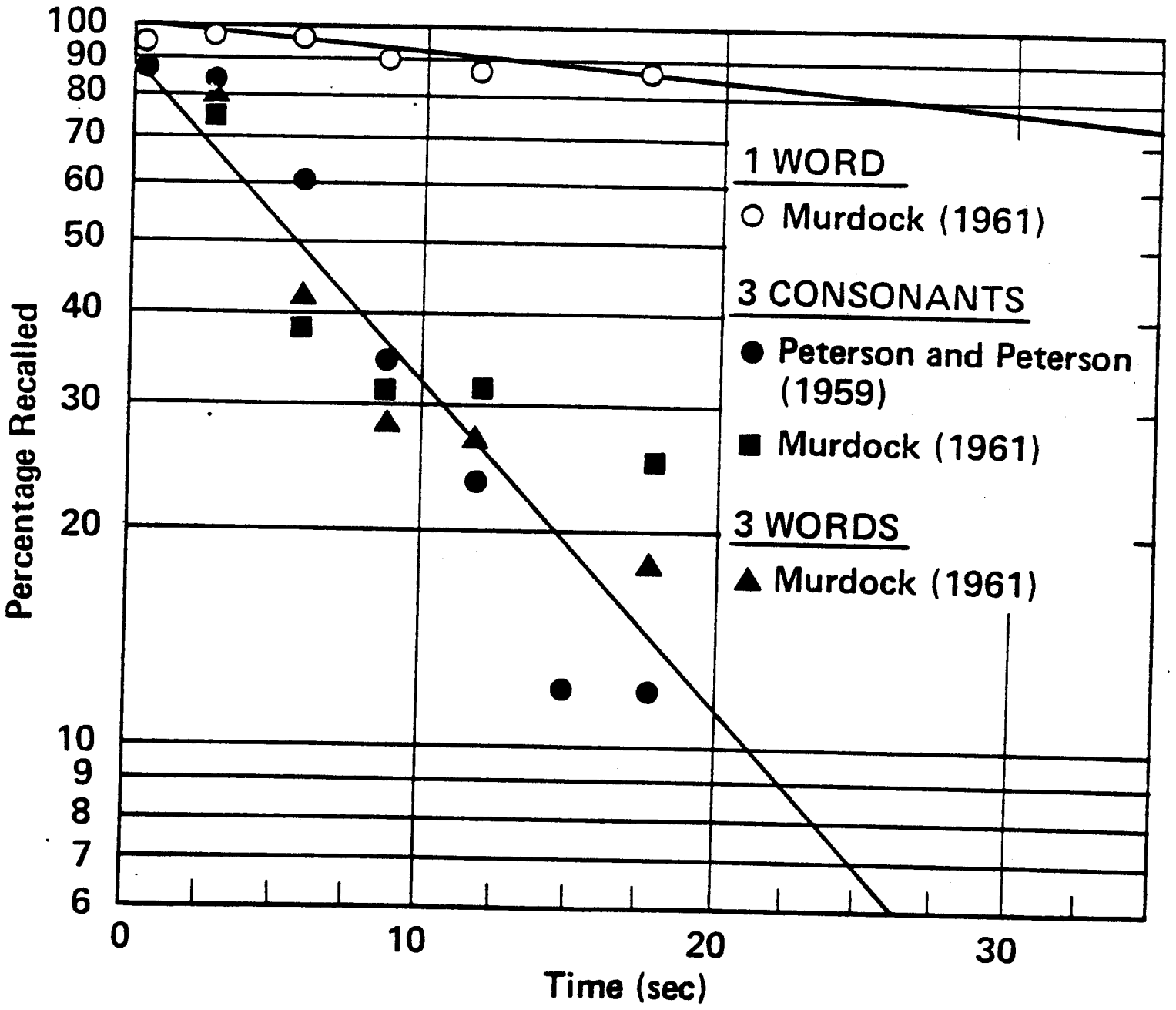

100
90
80
70
60
50
40
30
20
10
9
8
7
6
Percentage Recalled
0
10
20
30
Time (sec)
1 WORD
○ Murdock (1961)
3 CONSONANTS
● Peterson and Peterson (1959)
■ Murdock (1961)
3 WORDS
▲ Murdock (1961)

Fig. 7.

## COGNITIVE PROCESSOR RATE: $T_C$

| | |
|---|---|
| MATCHING ITEMS | 27-93 msec/item |
| DIGITS | 33 [27 - 39] msec/item |
| LETTERS | 40 [24 - 65] msec/item |
| WORDS | 47 [36 - 52] msec/item |
| GEOMETRICAL SHAPES | 50 msec/item |
| RANDOM FORMS | 68 [42 - 93] msec/item |
| NONSENSE SYLLABLES | 73 msec/item |
| COUNTING FEW OBJECTS | 40 - 172 msec/item |
| DOT PATTERNS | 46 msec/item |
| 3-D SHAPES | 94 [40 - 172] msec/item |
| PERCEPTUAL JUDGEMENT | 92 msec/inspection |
| CHOICE REACTION TIME | 153 msec/bit |
| SILENT COUNTING | 167 msec/digit |

**Fig. 8.**

*Variable Perceptual Processor Rate.*

More intense stimuli reduce the cycle time of the Perceptual Processor.

*Encoding Specificity Principle.*

To retrieve an item from Long-Term Memory, the user has to use the links that were encoded at the time the itemwas stored.

*Discimination Principle.*

The more similar things in memory are, the harder it is to recover them.

*Variable Cognitive Processor Rate.*

The Cognitive Processor cycle time can be reduced somewhat with effort or practice.

*Fitts's Law.*

The time to move the hand to a target is proportional to the log of the ratio of the distance to the target and target size.

*Power Law of Practice.*

When one plots learning curves on log-log paper, they tend to give a straight line.

*Uncertainty Principle.*

The more uncertainty in an action, the greater the reaction time.

*Ratinality Principle.*

The way to predict peoples' behavior in task-oriented situations is to understand their goals. The more one can understand, the better one can predict.

*Problem Space Principle.*

Goal-oriented behavior can be described in terms of a small number of operations performed over and over by the user.

## Sample Calculations

Finally, let us consider an example of the Model Human Processor description of the computer user in action.

> PROBLEM. When a symbol appears on a CRT, a user is to press a key.
> What is the time between signal and response.

SOLUTION. The user sees the signal and processes it with his Perceptual Processor. This requires a nominal 100 msec and results in a physical code for the symbol in the Visual Image Store and a symbolic code in Working Memory. The code in Working Memory requires one 70 msec cycle of the Cognitive Processor to trigger the response of pushing the response key. Pushing

PRINCIPLES OF OPERATION

0. RECOGNIZE-ACT COGNITIVE PROCESSOR CYCLE

1. VARIABLE PERCEPTUAL PROCESSOR RATE

2. ENCODING SPECIFICITY PRINCIPLE

3. DISCRIMINATION PRINCIPLE

4. VARIABLE COGNITIVE PROCESSOR RATE

5. FITTS'S LAW OF MOVEMENT TIMES

6. POWER LAW OF PRACTICE

7. THE UNCERTAINTY PRINCIPLE

8. RATIONALITY PRINCIPLE

9. PROBLEM SPACE PRINCIPLE

**Fig. 9.**

the key requires one 70 msec cycle of the motor processor to carry out. The entire action requires a nominal

$$\tau_P + \tau_C + \tau_M = 100 \text{ msec} + 70 \text{ msec} + 70 \text{ msec} = 240 \text{ msec.}$$

This is, in fact, a typical response time. The value can be recalculated using the upper and lower bounds for the parameters to give a range of times that includes most experiments.

> PROBLEM. A second symbol occurs just after the first. The user is to push the key if the second symbol is identical to the first. What is the time between signal and response?

SOLUTION. This just adds one Cognitive Processor cycle to determine if the signals are identical. The total time will now be

$$\tau_P + 2\tau_C + \tau_M = 100 \text{ msec} + 2(70 \text{ msec}) + 70 \text{ msec} = 310 \text{ msec.}$$

> PROBLEM. A programmer is programming a video game version of billiards. Frequently during the game one ball will bump into another causing the two balls to change speed and direction. How much time is available after the collision to compute the trajectories of the balls before they must be moved to preserve the illusion of causality?

SOLUTION. Movement must occur within one cycle of the Perceptual Processor, that is, within about 100 msec to have collision and subsequent movement appear part of the same event. Of course 100 msec is the time at which the illusion breaks down. To be sure we should use the lower bound of 50 msec. · An experiment by Michotte (1946/1963) shows that these numbers are approximately correct. Michotte showed people animated sequences of balls colliding with varying delays between when the first ball hit the second and the second ball moved. He asked them to judge whether the first ball caused the second one to move. If the delay was less than about 100 msec, people said the causality was immediate; if greater than 100 msec, they said the movement of the balls were independent events. If the collision was around 100 msec (from about 50 msec to about 150 msec) sometimes people said that the first ball had caused the second to move but that the collision was "sticky." Hence, if the balls are moved in less than 50 msec the illusion would remain in tact.

This completes the first lecture on user-computer interaction and cognitive psychology. The point can be stated thus: Most work in computer science concentrates on the machine side of what is really a two-sided relationship between man and machine. Yet it will likely be difficult to understand the human side from the use of such techniques as comparative experiments alone. What would seem to be required is what might be called a *model-based human factors* in which analytical models of performance capable of calculation and prediction are used to understand and drive experiments. The Model Human Processor is at the lowest level of these, summarizing the basic capabilities of the user. In the next lecture, we will give examples of how engineering calculations relevant to the design of human-computer interfaces can be built upon this base.

LECTURE 2

# EXAMPLES OF A MODEL-BASED HUMAN FACTORS

In the last lecture, we presented a model, the Model Human Processor, intended to summarize the most important limitations and capablities of humans. We shall now proceed to more extensive examples in the human-computer interaction domain.

## Pointing Devices

Let us begin by analyzing the pointing devices to be used with a display. Our first consideration is how long it takes the hand itself to move to some target.

PROBLEM. How long does it take to move the hand to a target $S$ cm wide whose center is $D$ cm distant? (See Fig. 10.)

SOLUTION. According to the Model Human Processor, the hand will make a series of micromovements to the target. Each micromovement will attempt to hit the target, but there will be some error each time. The time to reach the target will therefore be



$$T = nT_{step}$$

where $T$ is the total time, $T_{step}$ is the time for each micromovement, and $n$ is the number of steps required.

Let $X_i$ = be the distance remaining to the target after micromovment$_i$. Assume a constant error $\varepsilon$ for each micromovement. Then,

$$X_0 = D$$

$$X_1 = \varepsilon X_0 = \varepsilon D$$

$$X_2 = \varepsilon X_1 = \varepsilon(\varepsilon D) = \varepsilon^2 D$$

The hand stops when it is within the target, or

$$X_n = \varepsilon^n D = \frac{S}{2}$$

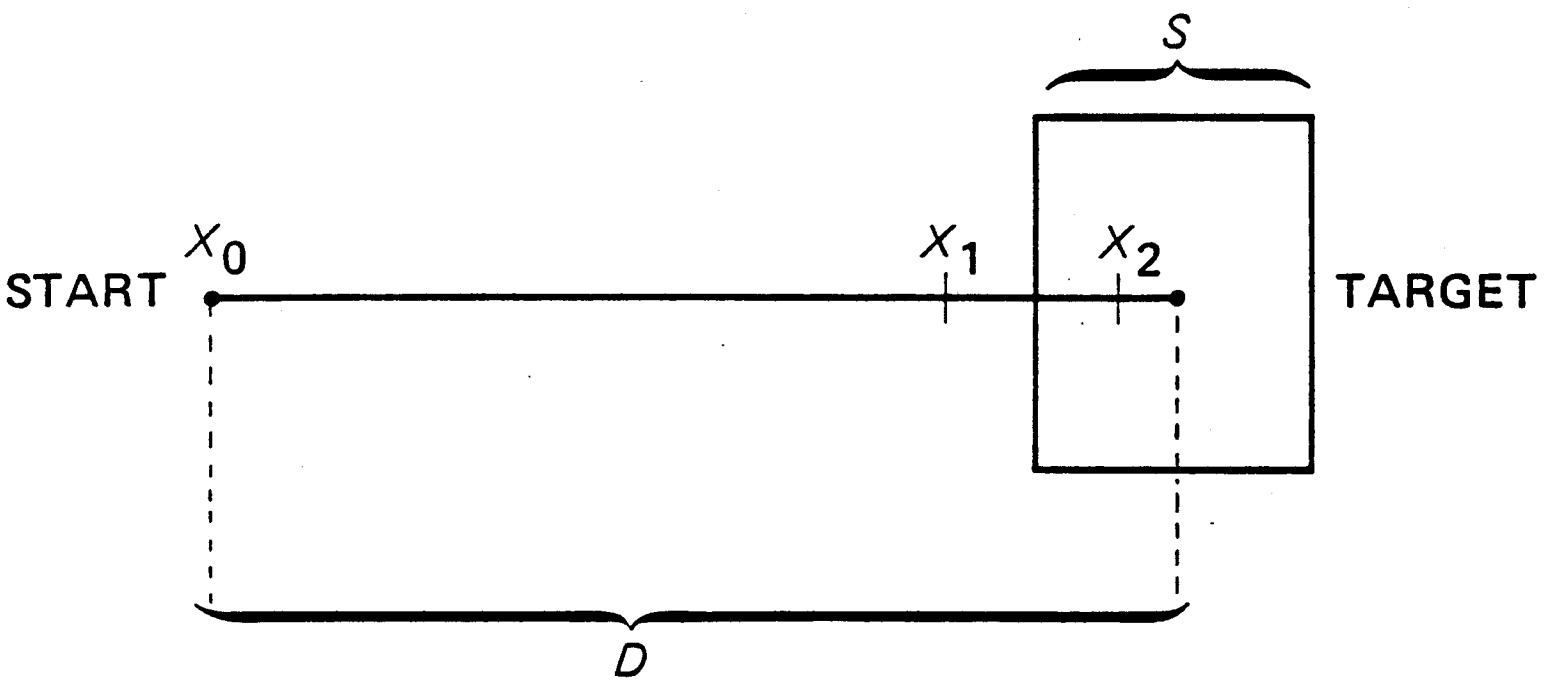



**Fig. 10.**

Solving for $n$ gives:

$$n = -\frac{\log_2\left[\frac{2D}{S}\right]}{\log_2\varepsilon}.$$

We use this last term to substitute for $n$ in Eq. (1) to obtain,

(2)

$$T = K\log_2\left[\frac{2D}{S}\right],$$

$$\text{where } K = \frac{-T_{step}}{\log_2\varepsilon}.$$

Eq. (2) is known as Fitts's Law. We can estimate the constants from first principles:

$$\varepsilon = \text{absolute discrimination} = 1/7$$

$$T_{step} = \tau_P + \tau_C + \tau_M = 240 \text{ msec}$$

to obtain a value for $K$ of

$$K = -240/-2.8 = 86 \text{ msec/bit} \quad \text{[from model]}.$$

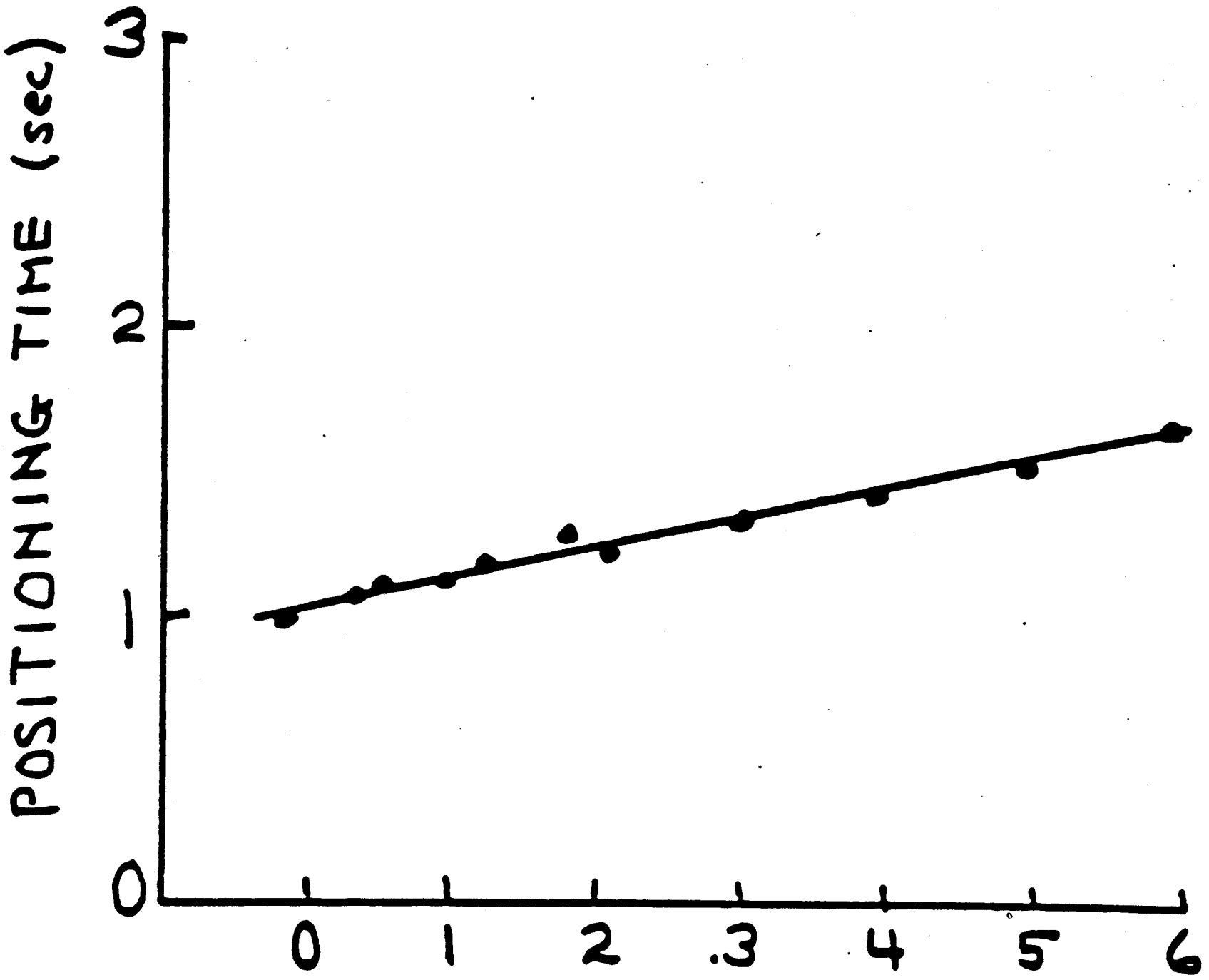
This is similar to the values obtained empirically in the literature:

(3)

$$K = 100\,[50\sim120]\text{ msec/bit} \quad \text{[from literature]}.$$

We now plot data (Fig. 11) from an experiment run to measure the time to point to targets of different distances and sizes with the "mouse" pointing device (a small box that rolls on a table to move a cursor). The pointing time for the mouse is plotted as a function of $\log_2(D/S + .5)$, a slight variant of the Fitts's Law expression in Eq. (2) that adds a small correction factor inside the log (Wellford, 1968). It can be seen that, indeed, the mouse pointing data is well fit by Fitts's Law. In this particular experiment (Card, English, and Burr, 1978), the mouse was compared to other devices: a joystick, step keys, entity keys. The mouse was best. The normal thing is just to make this comparison in line with the method of comparative experiments in human factors illustrated in the first lecture. But such comparisons are dangerous. If we do not know *why* the results come

MOUSE POSITIONING DATA




$$\log_2(D/S + .5)$$

$$T = 1.03 + .096 \log_2(D/S + .5)$$

Fig. 11.

out the way they do when we attempt to apply them in a new design, the results may not generalize.

Because we have a model and not just a comparative experiment, we can go farther. We know that the reason positioning time varies with size and distance is that positioning time for the mouse follows Fitts's Law. We now take note that the coefficient for Fitts's Law in Fig. 11 is about the same as the .1 sec/bit coefficient that derives from numerous tasks in the literature (Eq. 3) as well as our derivation from the Model Human Processor for hand pointing. This means that the limitation on pointing speed is in the human eye-hand system, not in the mouse. This in turn means that we are unlikely to make another device that will be faster (at least with the same muscles). Thus, using a model-based human factors we not only have more confidence in our results, we can also gain insight that helps us understand whether there are opportunities of improving the engineering of our subject device.

## Maximum Mouse Velocity

PROBLEM. A manufacturer wishes to build a system with a mouse. It would be both convenient and inexpensive if the system need cope with cursor speeds no greater than 50 cm/sec. Would that be enough?

SOLUTION. To solve this problem, we compute the average mouse velocity for the cycle on which it will be the maximum. That should be the first cycle, because the time is constant for each step and the largest distance is traveled on the first step:

$$V_{max} = \frac{X_0 - X_1}{\tau_P + \tau_C + \tau_M}$$

$$= \frac{D - \varepsilon D}{\tau_P + \tau_C + \tau_M}$$

$$= \left[\frac{1-\varepsilon}{\tau_P + \tau_C + \tau_M}\right] D$$

When we substitute for the constants we get,

$$V_{max} = 4.9D \text{ cm/sec.}$$

That is, to find the maximum velocity of the cursor on the screen, take the maximum dustance moved (in cm) and multiply it by 4.9 to get the velocity in cm/sec. The longest distance to move the cursor is diagonally from one corner to another. The screen being considered had a diagonal of 35 cm. Hence, the fastest velocity expected would be,

$$V_{max} = (4.9)(35)$$

$$= 171 \text{ cm/sec}.$$

This number is more than a factor of three greater than the 50 cm/sec considered. The validity of this calculation has been checked and is in agreement with empirical results (Card, Moran, and Newell, 1983).

## Text Editing

Now let us consider the analysis of a different sort of system: computer-based text editors. In this case, we begin with an application of the Rationality Principle from the Model Human Processor.

> PROBLEM. How long does it take an expert to modify text with a computer starting from a manuscript with marked modifications?

SOLUTION. The user will produce a goal-oriented method. We can analyze the user's behavior in terms of Goals, Operators, Methods available for meeting the goals, and Selection rules for deciding among alternative methods for the same goal. Such an analysis we call a GOMS analysis after the components in it. For example, a single editing task on the line-oriented editor POET looks like this:

```
GOAL: EDIT-MANUSCRIPT
    GOAL: EDIT-UNIT-TASK                    repeat until no more unit tasks
        GOAL: ACQUIRE-UNIT-TASK             if task not remembered
            GET-NEXT-PAGE                   if at end of manuscript
            GET-NEXT-TASK
        GOAL: EXECUTE-UNIT-TASK             if an edit task was found
            [select USE-QS-METHOD USE-LF-METHOD]
            GOAL: MODIFY-TEXT
```

[select USE-S-COMMAND USE-M-COMMAND]

VERIFY-EDIT

Roughly, this notation says that that the user breaks up the task of editing the manuscript into subgoals of editing a unit task (more or less a single modification). To accomplish each of these he first acquires the task by turning to the marked-up manuscript then executes the task acquired. To execute the task, he has a selection among different methods. Finally, he verifies that the edit is correct. Models of this sort can be drawn in outline, as above, or in extended detail. They can be used to predict sequences of user actions and times for editing particular manuscripts. (See Card, Moran, and Newell, 1983). When tested against real data they capture a reasonable first-order approximation of editing behavior.

In the limit, as the user becomes more expert, we can give a more simplified description consisting of the basic motor operations plus some small amont of mental operation. Because this model has the fewest number of operators when given at the keystroke level of operation (other levels of aggregation are possible), it is called the Keystroke-Level Model. The operators of the Keystroke-Level Model are summarized in Fig. 12. Notice that we still need an operator for mental activity *M* because even with practice and expertise, not all of the perceptual and cognitive activity is overlapped by motor operations. The rules in Fig. 12 predicting the location of *M* operations can be reduced to the proposition that whatever operations can exist together in memory as a chunk will have their cognitive operations overlapped by motor operations, but extra time (coded as *M* operators) will be required between chunks.

As an example, we give an analysis using the Keystroke-Level Model for an a display-oriented text-editor that uses the mouse.

> PROBLEM. How long does it take to correct a mistyped word typed *n* words previously.

SOLUTION. In this particular editor, the correction is accomplished by typing control-W to erase words back from the end until the defective word is reached, retyping it, then retyping the text erased. We can list each of the steps involved and code them in terms of the operators of the Keystroke-Level Model as follows:

**Method W:**

| | | |
|---|---|---|
| 1. | Press and hold CONTROL key | M K[CONTROL] |
| 2. | Invoke Backword *n* times | *n*(.25M K[W]) |

## THE KEYSTROKE MODEL

$T_{total}$ = Sum of $T_{unit\ task}$

$T_{unit\ task} = T_{acquisition} + T_{execute}$

OPERATORS

| | | |
|---|---|---|
| K[text] | KEY-IN text | $T_K$ = .20 sec |
| P[object] | POINT-TO object | $T_P$ = 1.10 sec |
| H[device] | HANDS-TO device | $T_H$ = .40 sec |
| M | MENTAL prepare | $T_M$ = 1.35 sec |
| R(delay) | WAIT FOR delay | |

RULES FOR LOCATION OF M OPERATOR

M only before decision point which cannot be chunked

1. First letter of a command name: M K[r]

2. First terminator of a variable argument string
K[word] M K[ESC], M K[ESC ESC] (1st terminator only)

3. Pointing to a command: M P[QUIT] K[ # 1]

Fig. 12.

| | |
|---|---|
| 3. Type new word | 5.5K[*word*] |
| 4. Retype destroyed text | 5.5($n$ – 1)K |

$$T_{execute} = (1 + .25n)t_M + (1 + 6.5n)t_K$$
$$= 1.6 + 2.16\,n \text{ sec}$$

This simplified model is a reasonably good predictor of performance time for experts (Fig. 13) and given that it is known what the sequence of operations will be. But when one is designing a system, that is just what one is designing.

This editor also comes with another method for accomplishing the same goal: The user stops typing, reaches for the mouse and points to the bad word, replaces it, then resumes typing. We can use the Keystroke-level Model to analyze this method giving us a basis on which to compre the two methods:

**Method R:**

| | |
|---|---|
| 1. TERMINATE TYPE IN | MK[ESC] |
| 2. REPLACE BAD WORD | |
| 2.1 Home hand onto mouse | H[MOUSE] |
| 2.2 Point to target word | P[*word*] |
| 2.3 Select word | K[MOUSEBUTTON2] |
| 2.4 Home hand onto keyboard | H[KEYBOARD] |
| 2.5 Invoke Replace command | MK[r] |
| 2.6 Type new word | 4.5K[*word*] |
| 2.7 Terminate Replace | MK[ESC] |
| 3.RESUME TYPING | |
| 3.1 Home hand onto mouse | H[MOUSE] |
| 3.2 Point to last input word | P[*word*] |
| 3.3 Select it | K[MOUSEBUTTON2] |
| 4. RE-ENTER TYPE-IN MODE | |
| 4.1 Home hand onto keyboard | H[KEYBOARD] |
| 4.2 Invoke Insert command | MK[i] |

# KEYSTROKE MODEL DATA

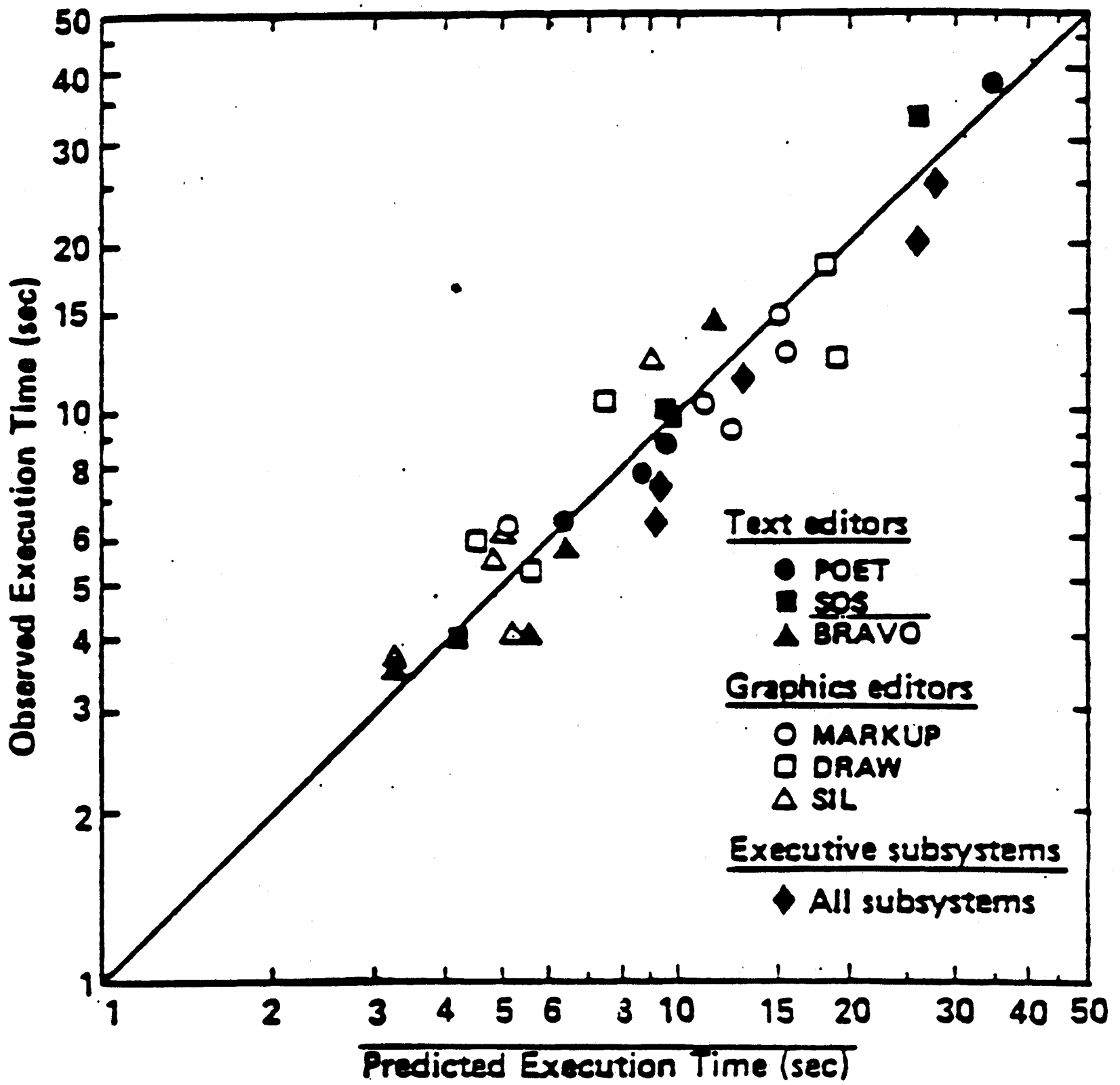



Fig. 13.

$$T_{execute} = 4t_M + 10.5t_K + 4t_H + 2t_P$$
$$= 1.6 + 2.16\,n \text{ sec}$$

Because we have a model of the interaction we can make a parametric comparison (solid lines, Fig. 14). We see that there is a region of the parameter $n$ in which each method is superior. Of course, we have ignored the relative frequencies with which users are likely to make each correct words for each value of $n$. Adding that to our analysis would be straightforward, but would complicate the presentation.

Now suppose we wanted to build a new command that would allow the user to move a cursor back to the word to be altered and then skip forward again without destroying the good text. In this new method, the user will type CONTROL-S to move the cursor back to the word to be changed and CONTROL-R to resume. The question is, can we tell before implementing anything about the usefulness of this command? Again we do an analysis:

**Method S:**

| | | |
|---|---|---|
| 1. | Setup Backskip command | M K[CONTROL] |
| 2. | Execute Backskip $n$ – 1 times | $(n - 1)$(.25M K[s]) |
| 3. | Call Backward command | M K[w] |
| 4. | Type new word | 4.5K[*word*] |
| 5. | Call Resume command | M K[CONTROL r] |

$$T_{execute} = (3 + (n-1))\,t_M + (n + 7.5)\,t_K$$
$$= 5.8\,t_M + .62n\,t_K$$

From Fig. 14(dotted line), we can see that there is a range in which this new method, Method S, is superior to the other two methods. Of course, in the design of the whole system we still have to worry whether the addition of a new command makes the system harder to learn or more prone to errors, but at least on grounds of time alone we can say the results are encouraging. Because we have a model to manipulate, we are in a position to explore consequences of designs analytically and to understand benchmark experiments.

We could carry the analysis into a number of other areas (see Card, Moran, and Newell, 1983 for some of these), but these examples should suffice to give the general idea. Our point was to illustrate how a model-based approach could help us to make progress on the human side of computer science. We first used a simplified engineering model of the human to summarize basic

# COMPARISON OF METHODS W, R AND S

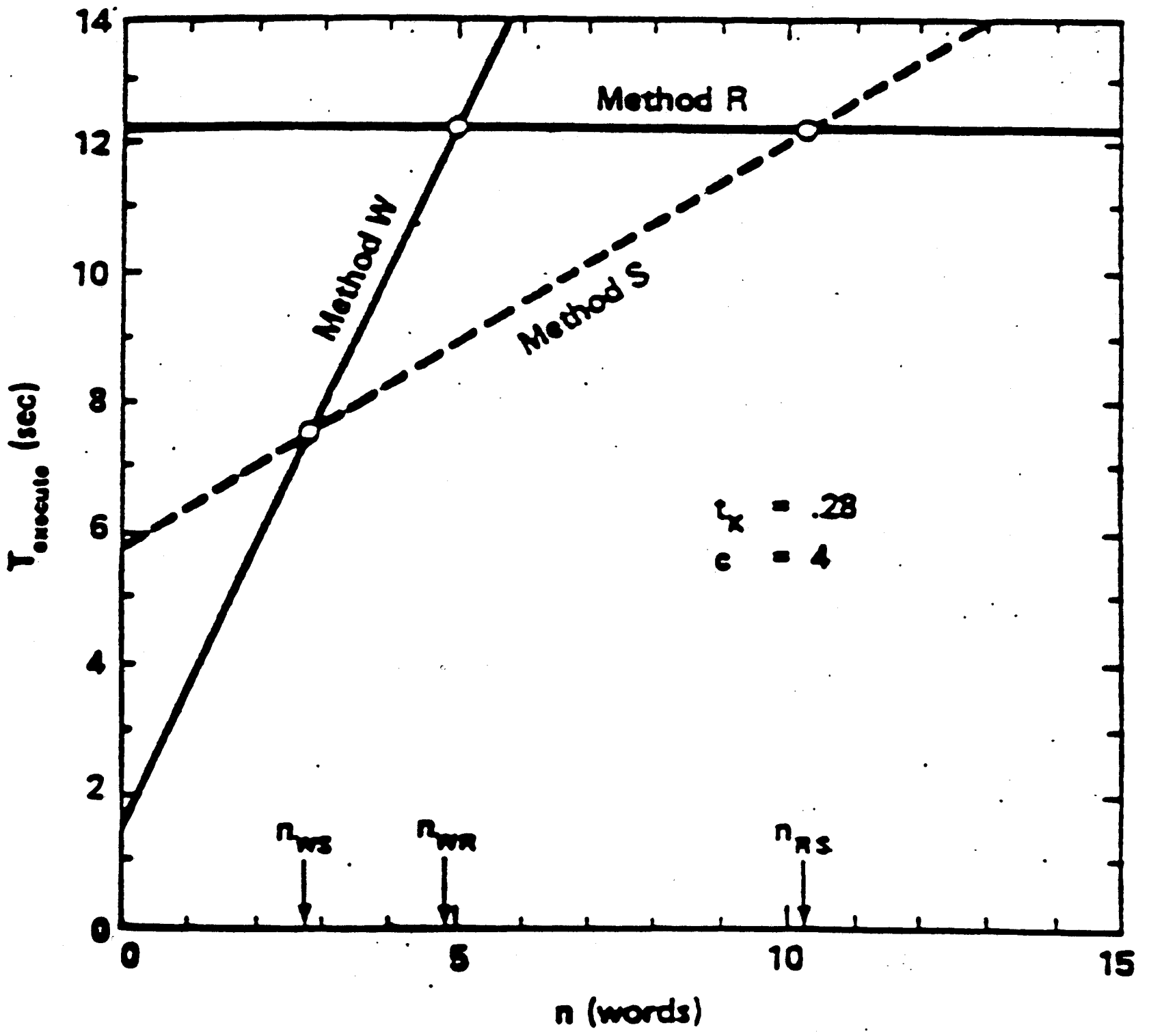



Fig. 14.

human capabilities. We then pushed into other models, the Fitts's Law model of the mouse and the Keystroke-Level Model of expert system interaction that allowed us to pursue calculations and understanding at a level above that of the basic Model Human Processor model. There are issues that are beyond the beginings of this framework but that we are hopeful can be brought within it with more research: the use of visual displays, for example, and the behavior of novice and casual users. But our theme has been that if ergonomic/human-factors knowledge is model or theory-based, it is easier to be put to analytic use. This then is our vision for how the human part of computer science might be developed. In the final lecture, we shall consider explicitly the question of how one might fit such knowledge as it develops into the computer science curriculum.

LECTURE 3

# HUMAN-COMPUTER INTERACTION IN THE COMPUTER SCIENCE CURRICULUM

In this final lecture we shall consider three things:

1. *Why* computer scientists need to know about human-computer interaction,
2. *Where* it fits in the ciurriculum, and
3. *What* might be taught.

## Why does a computer scientist need to know about human-computer interaction?

This question is easy. It is difficult to escape the conclusion that computer interfaces are becoming an increasingly important part of computer systems for the reasons we listed in Lecture 1: the availabilty of more computer power, bitmapped graphics and other graphical hardware innovations, increased subtlety of computer interaction, the possibility of "intelligent" interfaces. Beyond these technical reasons, and partially because of them, organized groups of users (unions and user groups) guarantee that computer scientists are going to hear a lot about usability in the future. There may even be attempts at regulation, so the computer scientists of the future would do well to be informed. But the real reason is that greater understanding of human-computer interaction will be necessary for the computer scientist to do his work.

## Where does human-computer interaction fit in the curriculum?

Let us rephrase this question as a set of nine computer science curriculum design issues:

1. *Should human-computer interaction be in the computer science curriculum?*

Our answer, perhaps not surprisingly, would be yes, that a number of topics in computer science are heavily influenced both by the structure of computers and by the structure of humans, that currently virtually all the study is on the computer side, and that this should be balanced by some on the human side.

2. *Is it peripheral or central to computer science?*

Examples of peripheral topics would be social responsibility or databases. Human-computer

interaction belongs first on the periphery as the subject of graduate seminars and small parts of courses, moving in towards the core as it gains adequate content.

*3. Should it be oriented towards evaluating systems or designing systems?*

Our choice is design. Design is a key activity in computer science. Design time is the only time in which there is enough leverage to really make a large difference. This implies an emphasis on *theory*, when available; *tabulated facts*, when usable; *guidelines*, when possible; and *design methodologies*, again, when usable.

*4. Who should do the human engineering? Human factors specialists or computer scientists?*

The emphasis on design implies that the computer scientist is going to be the major player in the human engineering of computing systems. Interface considerations must be traded off against other factors such as machine speed, memory, display technology, and data structures. An example (due to Robert Sproull) is when a designer of a drawing system chooses between using a geometric model or a pen model. This choice of abstraction by itself determines that some tasks will be easier and some tasks more difficult; it determines whether the storage requirements will be large or small; it determines some of the errors users will have and some of the difficulties there will be in training. These issues must be traded against each other directly. By the time the designer has made this decision, the major interface decisions have already been made.

Another reason computer scientists will be the ones to do human engineering is that it is the computer scientists who will be the first ones to see/invent many technology opportunities. A related reason is that much human engineering may get done through the creation of user interface software packages. The creation of these is a computer science task.

Finally, the history of having human factors specialists in other engineering domains shows they tend to play minor roles and have little impact on the overall system. (They get to choose the fonts, the color of the case, and to write the instructions).

Of course I have talked as if people came in certain stereotyped packages. In fact, we are beginning to see more students who have acquire respectable backgrounds in both computer science and the relevant psychology as well as professionals who have cross experience. This should warn us to judge research contributions directly, rather than by primary professional affiliation. A related qualification is the real lack of a laboratory culture in computer science. This means that while computer scientists are preeminant in design, they often do not do particularly well when it comes to making measurements and working out theories of user behavior. If computer scientists are going to do *reasearch* on human-computer interaction as well as building

systems, it would be nice to ensure in their training that they have some background in laboratory science, including at least some in the natural sciences.

5. *At what course-level should human-computer interaction enter the curriculum?*

Ideally, human-computer interaction would enter the curriculum at the undergraduate level for those things that can be established. Presently, however, the most appropriate place for most of the material in this area is in graduate seminars.

6. *Should what is taught be human-computer interaction or human factors?*

We think it should be human-computer interaction. The issues are closely bound up with software issues. Splitting off of the software issues and leaving only the human factors considerations is likely to result in the isolation and ignoring of human factors by computer scientists.

7. *Should the sort of material in these lectures be a course or part of one?*

For similar reasons as above, we think it preferable that this material be part of a course on the interface. The human part of programming languages should go with the programming language courses, where appropriate, and the human part of interfaces should go with interactive systems and the human part of graphics should go with graphics. This especially should be a consideration for text-book writers.

8. *Should all the human related aspects be in one course or separate courses?*

This is a version of the previous question. The human-related aspects of computer science ought to be separated into different courses because the focus in computer science is not on humans per se; it is on whatever (human aspects included) is relevant to the various topics of computing.

9. *What should be the objectives of such a course (or part of one)?*

The objectives of the part of the course that teaches about human factors should be: (1) to establish the goal of taking the user into account; (This has been the lament in the human factors discipline all these years.) (2) to give students a realistic operational picutre of the user; and (3) to give the students some techniques to use in design and research.

There is one additional comment that should be made. The sorts of interfaces that are becoming possible and that are much in need of university-based research depend on modern hardware. Yet the equipment available to most users in universities is such as to make training and research on modern interfaces almost impossible. Students trained on interfaces using 80 character x 24 line displays running on a time-sharing mainframe will be largely unaware of and unprepared for the rich possibilities for modern interfaces. *Probably the most cost-effective expenditure that any government funding agency could make toward raising the general level of the state-of-the-art in human-computer interaction would be to reequip the universities through equipment grants.*

## What would be the content of a course in interactive systems?

To explore this issue, let us propose a straw-man syllabus for a course in interactive systems to see where things might fit. Our suggestion is as follows:

1. *Overall interaction model.*

Computer scientists will program not just the type of systems they commonly use at universities, but also devices such as aircraft flight systems, power plants, space telescopes, automated bank tellers, and other systems that have embedded computers. Students should be given an overall model, general enough to embrace all these contexts. The Sheridan model presented earlier has some attractions as does recent work by Rasmussen and others.

2. *Characterization of the human.*

Computer scientists need some approximate characterization of human capabilities to structure their knowledge of humans so as to be able to predict the gross reasonableness of various proposals. (This is an intended role of the Model Human Processor.) They also need to be taught what applied models there are for predicting user responses.

3. *Dialogue styles and task characteristics.*

Putting these all in order is one of the tasks in this area that needs tidying up. Students should have exposure to a systematically collected sample of styles, since at this stage in the field, concrete cases are one of the most important design influences. These, of course, are the building

blocks out of which the students will later build systems. Suggested topics include users' models, windows, menus, alphameric dialogues, and devices. The topics would also include some parts of computer graphics, data structures, and similar topics. But different interaction techniques are appropriate for different tasks, so this activities also leads to some taxonomizing of the tasks people actually do at the interface.

4. *Problems.*

There also needs to be exposure to the challenges in this area and the means that exist for fielding them. Examples: the integration problem, controlling detail on the display, increasing functionality without proportionally increasing complexity.

5. *User interface management systems.*

How can the complexity of programming the interface be reduced by making packages that support components of the interaction, much as is done for graphics? The supposed benefits of such packages include reduced costs of constructing interfaces, smoother interaction, higher reliability, and increased interface consistency across applications and within applications.

6. *Intelligent interfaces.*

Using an interface is a communicaiton process. Artificial intelligence techniques can be applied to the interface to produce different types of interfaces: For example interfaces can be made that coach their users by giving them hints (Sleeman and Brown, 1982). Interfaces can be built that tailor their interaction based on dynamic models of the user. Systems can be made that make use of the "conversational postulates" of human discourse.

## Conclusion

Let us recapitulate our theme:

Now (in the next five years) is the time to deal with the interface. There are finally adequate computational resources beyond the needs of the subject program such that we can afford to spend code and memory on the interface. A lot is at stake, since the usefulness of many programs depends critically on the interface.

We would suggest requirements for making progress in this area as follows: (1) The human side of human-computer interaction must be taken seriously. (2) This must happen within

computer science (not within just psychology or human factors). (3) It must happen by building an engineering theory of human behavior that encompasses task analysis, calculation, approximation, and is theory-based. These are necessary so that they can aid system building by computer scientists at design time.

We have outlined some progress in satisfying these requirements: The Model Human Processor as a base and models such as the GOMS model and the Keystroke-Level Model as examples. The Model Human Processor is as yet very crude. It gives some useful results, but there is much that is still missing in terms of its coverage. It is a synthesis of various results that are around, not a new theory on its own. It tries to make it possible for computer scientiststo work with a reasonable model of what the user is all about.

Finally, we have tried to show where this sort of effort would fit into a curriculum concerned with more than just the user per se.

## References


Averbach, E. and Coriell, A. S. (1961).

Short-term memory in vision. *Bell System Technical Journal 40*, 309–328.

Card, S. K.; English, W. K.; and Burr, B. J. (1978).

Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics 21*, 601–613.

Card, S. K.; Moran, T. P.; and Newell, A. N. (1983).

*The Psychology of Human-Computer Interaction.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Conrad, R. and Hull, A. J. (1968).

The preferred layout for numerical data-entry keysets. *Ergonomics 11*, 165–173.

Darwin, C. J.; Turvey, M. T.; and Crowder, R. G. (1972).

An auditory analogue of the Sperling partial report procedure: Evidence for brief auditory storage. *Cognitive Psychology 3*, 255–267.

Kurke, M. I. (1956).

Evaluation of displays incorporating quantitative and check-reading characteristics. *Journal of Applied Psychology 40*, 233–236.

Michotte, A. (1946/1963).

*The Perception of Causaility.* New York: Basic Books, 1963. Originally published as *La Perception de la Causalite.* Louvain: Publications Universitaires de Louvain, 1946.

Murdock, B. B. Jr. (1961).

Short-term retention of single paired-associates. *Psychological Reports 8*, 280.

National Research Council 1982).

Automation in Combat Aircraft. Washington, D. C.: National Academy Press.

National Research Council (1983).

Research Needs for Human Factors. Washington, D. C.: National Academy Press.

Peterson, L. R. and Peterson, M. J. (1959).

Short-term retention of individual verbal items. *Journal of Experimental Psychology 58*, 193–198.

Siewiorek, D.; Bell, G.; and Newell, A. (1981).

*Computer Structures.* New York: McGraw-Hill.

Sleeman, D. and Brown, J. S. , eds. (1982).

*Intelligent Tutoring Systems.* London: Academic Press, 1982.

Sperling, G. (1960).

The information available in brief visual presentations. *Psychological Monographs 74* (11, Whole No. 498).

Welford, A. T. (1968). *Fundamentals of Skill.* London: Methuen.