CHAPTER 3

Cognitive Engineering

DONALD **A. NORMAN**

PROLOGUE

cognitive Engineering, a term invented to reflect the enterprise I find myself engaged in: neither Cognitive Psychology, nor Cognitive Science, nor Human Factors. It is a type of applied Cognitive Science, trying to apply what is known from science to the design and construction of machines. It is a surprising business. On the one hand, there actually is quite a lot known in Cognitive Science that can be applied. But on the other hand, our lack of knowledge is appalling. On the one hand, computers are ridiculously difficult to use. On the other hand, many devices are difficult to use—the problem is not restricted to computers, there are fundamental difficulties in understanding and using most complex devices. So the goal of Cognitive Engineering is to come to understand the issues, to show how to make better choices when they exist, and to show what the tradeoffs are when, as is the usual case, an improvement in one domain leads to deficits in another.

In this chapter I address some of the problems of applications that have been of primary concern to me over the past few years and that have guided the selection of contributors and themes of this book. The chapter is not intended to be a coherent discourse on Cognitive Engineering. Instead, I discuss a few issues that seem central to the

way that people interact with machines. The goal is to determine what are the critical phenomena: The details can come later. Overall, I have two major goals:

1. To understand the fundamental principles behind human action and performance that are relevant for the development of engineering principles of design.

2. To devise systems that are pleasant to use—the goal is neither efficiency nor ease nor power, although these are all to be desired, but rather systems that are pleasant, even fun: to produce what Laurel calls "pleasurable engagement" (Chapter 4).

AN ANALYSIS OF TASK COMPLEXITY

Start with an elementary example: how a person performs a simple task, Suppose there are two variables to be controlled. How should we build a device to control these variables? The control question seems trivial: If there are two variables to be controlled, why not simply have two controls, one for each? What is the problem? It turns out that there is more to be considered than is obvious at first thought. Even the task of controlling a single variable by means of a single control mechanism raises a score of interesting issues.

One has only to watch a novice sailor attempt to steer a small boat to a compass course to appreciate how difficult it can be to use a single control mechanism (the tiller) to affect a single outcome (boat direction). The mapping from tiller motion to boat direction is the opposite of what novice sailors sometimes expect. And the mapping of compass movement to boat movement is similarly confusing. If the sailor attempts to control the boat by examining the compass, determining in which direction to move the boat, and only then moving the tiller, the task can be extremely difficult.

> *Experienced sailors will point out that this formulation puts the problem in its clumsiest, most difficult form: With the right formulation,* ***or*** *the right conceptual model, the task is not complex. That comment makes two points. First, the description I gave is a reasonable one for many novice sailors: The task is quite dijjjcult for them. The point is not that there are simpler ways of viewing the task, but that even a task that has but a single mechanism to control a single variable can be difficult to understand, to learn, and to do. Second, the comment reveals the power of the proper conceptual model of the*

situation: The correct conceptual model can transform confusing, difficult tasks into simple, straightforward ones. This is an important point that forms the theme of a later section.

Psychological Variables Differ From Physical Variables

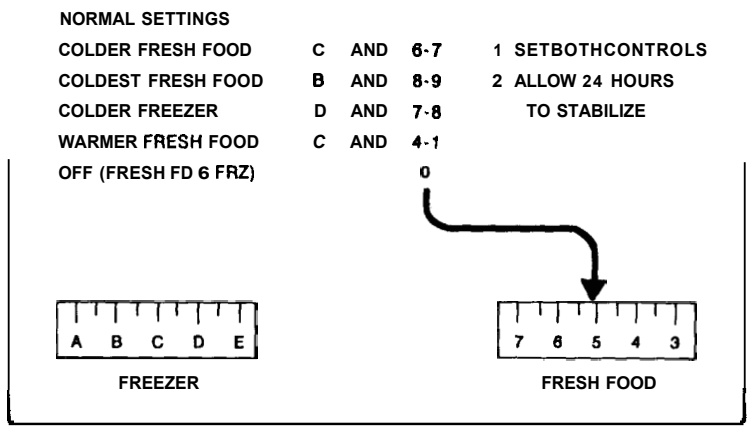
There is a discrepancy between the person's *psychologically* expressed goals and the *physical* controls and variables of the task. The person starts with goals and intentions. These are *psychological* variables. They exist in the mind of the person and they relate directly to the needs and concerns of the person. However, the task is to be performed on *aphysical* system, with physical mechanisms to be manipulated, resulting in changes to the physical variables and system state. Thus, the person must interpret the physical variables into terms relevant to the psychological goals and must translate the psychological intentions into physical actions upon the mechanisms. This means that there must be a stage of interpretation that relates physical and psychological variables, as well as functions that relate the manipulation of the physical variables to the resulting change in physical state.

In many situations the variables that can easily be controlled are not those that the person cares about. Consider the example of bathtub water control. The person wants to control rate of total water flow and temperature. But water arrives through two pipes: hot and cold. The easiest system to build has two faucets and two spouts. **As** a result, the physical mechanisms control rate of hot water and rate of cold water. Thus, the variables of interest to the user interact with the two physical variables: Rate of total flow is the sum of the two physical variables; temperature is a function of their difference (or ratio). The problems come from several sources:

1. *Mapping problems.* Which control is hot, which is cold? Which way should each control be turned to increase or decrease the flow? (Despite the appearance of universal standards for these mappings, there are sufficient variations in the standards, idiosyncratic layouts, and violations of expectations, that each new faucet poses potential problems.)
2. *Ease of control.* To make the water hotter while maintaining total rate constant requires simultaneous manipulation of both faucets.
3. *Evaluation.* With two spouts, it is sometimes difficult to determine if the correct outcome has been reached.

Faucet technology evolved to solve the problem. First, mixing spouts were devised that aided the evaluation problem. Then, "single control" faucets were devised that varied the psychological factors directly: One dimension of movement of the control affects rate of flow, another orthogonal dimension affects temperature. These controls are clearly superior to use. They still do have a mapping problem—knowing what kind of movement to which part of the mechanism controls which variable—and because the mechanism is no longer as visible as in the two-faucet case, they are not quite so easy to understand for the first-time user. Still, faucet design can be used as a positive example of how technology has responded to provide control over the variables of psychological interest rather than over the physical variables that are easier and more obvious.

It is surprisingly easy to find other examples of the two-control task. The water faucets is one example. The loudness and balance controls on some audio sets is another. The temperature controls of some refrigerator–freezer units is another. Let me examine this latter example, for it illustrates a few more issues that need to be considered, including the invisibility of the control mechanisms and a long time delay between adjustment of the control and the resulting change of temperature.



There are two variables of concern to the user: the temperature of the freezer compartment and the temperature of the regular "fresh

food" compartment. At first, this seems just like the water control example, but there is a difference. Consider the refrigerator that I own. It has two compartments, a freezer and a fresh foods one, and two controls, both located in the fresh foods section. One control is labeled "freezer," the other "fresh food," and there is an associated instruction plate (see the illustration). But what does each control do? What is the mapping between their settings and my goal? The labels seem clear enough, but if you read the "instructions" confusion can rapidly set in. Experience suggests that the action is not as labeled: The two controls interact with one another. The problems introduced by this example seem to exist at almost every level:

1. Matching the psychological variables of interest to the physical variables being controlled. Although the labels on the control mechanisms indicate some relationship to the desired psychological variables, in fact, they do not control those variables directly.

2. The mapping relationships. There is clearly strong interaction between the two controls, making simple mapping between control and function or control and outcome difficult.

3. Feedback. Very slow, so that by the time one is able to determine the result of an action, so much time has passed that the action is no longer remembered, making "correction" of the action difficult.

4. Conceptual model. None. The instructions seem deliberately opaque and nondescriptive of the actual operations.

I suspect that this problem results from the way this refrigerator's cooling mechanism is constructed. The two variables of psychological interest cannot be controlled directly. Instead, there is only one cooling mechanism and one thermostat, which therefore, must be located in either the "fresh food' section or in the freezer, but not both. A good description of this mechanism, stating which control affected which function would probably make matters workable. If one mechanism were clearly shown to control the thermostat and the other to control the relative proportion of cold air directed toward the freezer and fresh foods section, the task would be much easier. The user would be able to get a clear conceptual model of the operation. Without a conceptual model, with a 24-hour delay between setting the controls and determining the results, it is almost impossible to determine how to operate the controls. Two variables: two controls. Who *could believe that it would be so difficult?*

Even Simple Tasks Involve a Large Number of Aspects

The conclusion to draw from these examples is that even with two variables, the number of aspects that must be considered is surprisingly large. Thus, suppose the person has two psychological goals, G_1 and G_2. These give rise to two intentions, I_1 and I_2, to satisfy the goals. The system has some physical state, S, realized through the values of its variables: For convenience, let there be two variables of interest, V_1 and V_2. And let there be two mechanisms that control the system, M_1 and M_2. So we have the psychological goals and intentions (G and I) and the physical state, mechanisms, and variables (S, M, and V). First, the person must examine the current system state, S, and evaluate it with respect to the goals, G. This requires translating the physical state of the system into a form consistent with the psychological goal. Thus, in the case of steering a boat, the goal is to reach some target, but the physical state is the numerical compass heading. In writing a paper, the goal may be a particular appearance of the manuscript, but the physical state may be the presence of formatting commands in the midst of the text. The difference between desired goal and current state gives rise to an intention, again stated in psychological terms. This must get translated into an action sequence, the specification of what physical acts will be performed upon the mechanisms of the system. To go from intention to action specification requires consideration of the mapping between physical mechanisms and system state, and between system state and the resulting psychological interpretation. There may not be a simple mapping between the mechanisms and the resulting physical variables, nor between the physical variables and the resulting psychological states. Thus, each physical variable might be affected by an interaction of the control mechanisms: $V_1 = f(M_1, M_2)$ and $V_2 = g(M_1, M_2)$. In turn, the system state, S is a function of all its variables: $S = h(V_1, V_2)$. And finally, the mapping between system state and psychological interpretation is complex. All in all, the two variable–two mechanism situation can involve a surprising number of aspects. The list of aspects is shown and defined in Table 3.1.

TABLE 3.1
ASPECTS OF A TASK

Aspect	Description
Goals and intentions.	A goal is the state the person wishes to achieve; an intention is the decision to act so as to achieve the goal.
Specification of the action sequence.	The psychological process of determining the psychological representation of the actions that are to be executed by the user on the mechanisms of the system.
Mapping from psychological goals and intentions to action sequence.	In order to specify the action sequence, the user must translate the psychological goals and intentions into the desired system state, then determine what settings of the control mechanisms will yield that state, and then determine what physical manipulations of the mechanisms are required. The result is the internal, mental specification of the actions that are to be executed.
Physical state of the system.	The physical state of the system, determined by the values of all its physical variables.
Control mechanisms.	The physical devices that control the physical variables.
Mapping between the physical mechanisms and system state.	The relationship between the settings of the mechanisms of the system and the system state.
Interpretation of system state.	The relationship between the physical state of the system and the psychological goals of the user can only be determined by first translating the physical state into psychological states (perception), then interpreting the perceived system state in terms of the psychological variables of interest.
Evaluating the outcome.	Evaluation of the system state requires comparing the interpretation of the perceived system state with the desired goals. This often leads to a new set of goals and intentions.

TOWARD A THEORY OF ACTION

It seems clear that we need to develop theoretical tools to understand what the user is doing. We need to know more about how people actually do things, which means a theory of action. There isn't any realistic hope of getting *the* theory of action, at least for a long time, but

certainly we should be able to develop approximate theories.[1] And that is what follows: an approximate theory for action which distinguishes among different stages of activities, not necessarily always used nor applied in that order, but different kinds of activities that appear to capture the critical aspects of doing things. The stages have proved to be useful in analyzing systems and in guiding design. The essential components of the theory have already been introduced in Table 3.1.

In the theory of action to be considered here, a person interacts with a system, in this case a computer. Recall that the person's goals are expressed in terms relevant to the person—in psychological terms—and the system's mechanisms and states are expressed in terms relative to it—in physical terms. The discrepancy between psychological and physical variables creates the major issues that must be addressed in the design, analysis, and use of systems. I represent the discrepancies as two gulfs that must be bridged: the *Gulf of Execution* and the *Gulf of Evaluation*, both shown in Figure 3.1.[2]

The Gulfs of Execution and Evaluation

The user of the system starts off with goals expressed in psychological terms. The system, however, presents its current state in physical terms. Goals and system state differ significantly in form and content, creating the Gulfs that need to be bridged if the system can be used (Figure 3.1). The Gulfs can be bridged by starting in either direction. The designer can bridge the Gulfs by starting at the system side and moving closer to the person by constructing the input and output characteristics of the interface so as to make better matches to the

[1] There is little prior work in psychology that can act as a guide. Some of the principles come from the study of servomechanisms and cybernetics. The first study known to me in psychology—and in many ways still the most important analysis—is the book *Plans and the Structure of Behavior* by Miller, Galanter, and Pribram (1960) early in the history of information processing psychology. Powers **(1973)** applied concepts from control theory to cognitive concerns. In the work most relevant to the study of Human-Computer Interaction, Card, Moran, and Newell (1983), analyzed the cycle of activities from Goal through Selection: the GOMS model (*Goal, Operator, Methods, Selection).* Their work is closely related to the approach given here. This is an issue that has concerned me for some time, so some of my own work is relevant: the analysis of errors, of typing, and of the attentional control of actions (Norman, **1981a, 1984b, 1986;** Norman & Shallice, **1985;** Rumelhart & Norman, **1982).**

[2] The emphasis on the the discrepancy between the user and the system, and the suggestion that we should conceive of the discrepancy as a Gulf that must be bridged by the user and the system designer, came from Jim Hollan and Ed Hutchins during one of the many revisions of the Direct Manipulation chapter (Chapter 5).

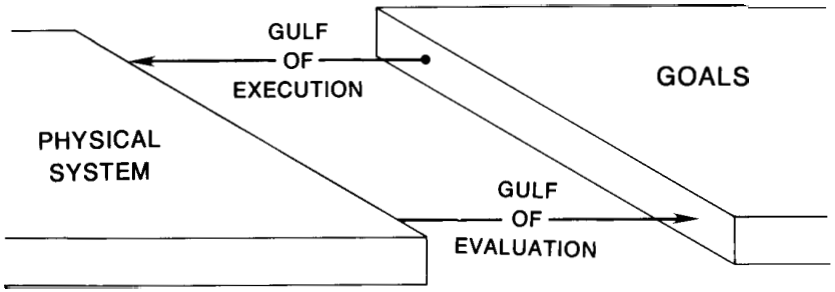


FIGURE 3.1. The Gulfs of Execution and Evaluation. Each Gulf is unidirectional: The Gulf of Execution goes from Goals to Physical System; the Gulf of Evaluation goes from Physical System to Goals.

psychological needs of the user. The user can bridge the Gulfs by creating plans, action sequences, and interpretations that move the normal description of the goals and intentions closer to the description required by the physical system (Figure 3.2).

Bridging the Gulf of Execution. The gap from goals to physical system is bridged in four segments: intention formation, specifying the action sequence, executing the action, and, finally, making contact with the input mechanisms of the interface. The intention is the first step, and it starts to bridge the gulf, in part because the interaction language demanded by the physical system comes to color the thoughts of the person, a point expanded upon in Chapter 5 by Hutchins, Hollan, and Norman. Specifying the action sequence is a nontrivial exercise in planning (see Riley & O'Malley, 1985). It is what Moran calls matching the internal specification to the external (Moran, 1983). In the terms of the aspects listed in Table 3.1, specifying the action requires translating the psychological goals of the intention into the changes to be made to the physical variables actually under control of the system. This, in turn, requires following the mapping between the psychological intentions and the physical actions permitted on the mechanisms of the system, as well as the mapping between the physical mechanisms and the resulting physical state variables, and between the physical state of the system and the psychological goals and intentions.

After an appropriate action sequence is determined, the actions must be executed. Execution is the first physical action in this sequence: Forming the goals and intentions and specifying the action sequence were all mental events, Execution of an action means to do something, whether it is just to say something or to perform a complex motor

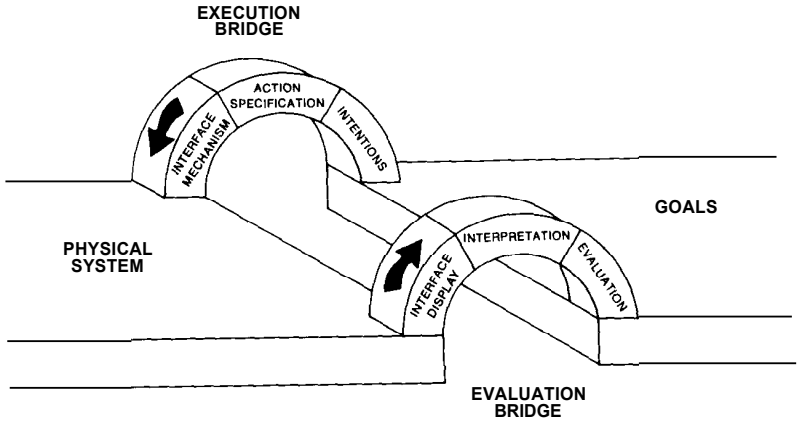


FIGURE 3.2. Bridging the Gulfs of Execution and Evaluation. The Gulf of *Execution* is bridged from the psychology side by the user's formation of intentions relevant to the system and the determination of an action sequence. It is bridged from the system side when the designer of the system builds the input characteristics of the interface. The Gulf of *Evaluation* is bridged from the psychology side by the user's perception of the system state and the interpretation placed on that perception, which is then evaluated by comparing it with the original goals and intentions. It is bridged from the system side when the designer builds the output characteristics of the interface.

sequence. Just what physical actions are required is determined by the choice of input devices on the system, and this can make a major difference in the usability of the system. Because some physical actions are more difficult than others, the choice of input devices can affect the selection of actions, which in turn affects how well the system matches with intentions. On the whole, theorists in this business tend to ignore the input devices, but in fact, the choice of input device can often make an important impact on the usability of a system. (See Chapter 15 by Buxton for a discussion of this frequently overlooked point.)

Bridging the Gulf of Evaluation. Evaluation requires comparing the interpretation of system state with the original goals and intentions. One problem is to determine what the system state is, a task that can be assisted by appropriate output displays by the system itself. The outcomes are likely to be expressed in terms of physical variables that bear complex relationships to the psychological variables of concern to the user and in which the intentions were formulated. The gap from system to user is bridged in four segments: starting with the output

displays of the interface, moving to the perceptual processing of those displays, to its interpretation, and finally, to the evaluation—the comparison of the interpretation of system state with the original goals and intention. But in doing all this, there is one more problem, one just beginning to be understood, and one not assisted by the usual forms of displays: the problem of level. There may be many levels of outcomes that must be matched with different levels of intentions (see Norman, 1981a; Rasmussen in press; Rasmussen & Lind, 1981). And, finally, if the change in system state does not occur immediately following the execution of the action sequence, the resulting delay can severely impede the process of evaluation, for the user may no longer remember the details of the intentions or the action sequence.

Stages of User Activities

A convenient summary of the analysis of tasks is is that the process of performing and evaluating an action can be approximated by seven stages of user activity' (Figure **3.3):**

- Establishing the Goal
- Forming the Intention
- Specifying the Action Sequence
- Executing the Action
- Perceiving the System State
- Interpreting the State
- Evaluating the System State with respect to the Goals and Intentions

3 The last two times I spoke of an approximate theory of action (Norman, 1984a, 1985) I spoke of four stages. Now I speak of seven. **An** explanation seems to be in order. The answer really is simple. The full theory of action is not yet in existence, but whatever its form, it involves a continuum of stages on both the action/execution side and the side. The notion of stages is a simplification of the underlying theory: I do not believe that there really are clean, separable stages. However, for practical application, approximating the activity into stages seems reasonable and useful. Just what division of stages should be made, however, seems less clear. In my original formulations, I suggested four stages: intention, action sequence, execution, and evaluation. In this chapter I separated goals and intentions and expanded the analysis of evaluation by adding perception and interpretation, thus making the stages of evaluation correspond better with the stages of execution: Perception is the evaluatory equivalent of execution, interpretation the equivalent of the action sequence, and evaluation the equivalent of forming the intention. The present formulation seems a richer, more satisfactory analysis.

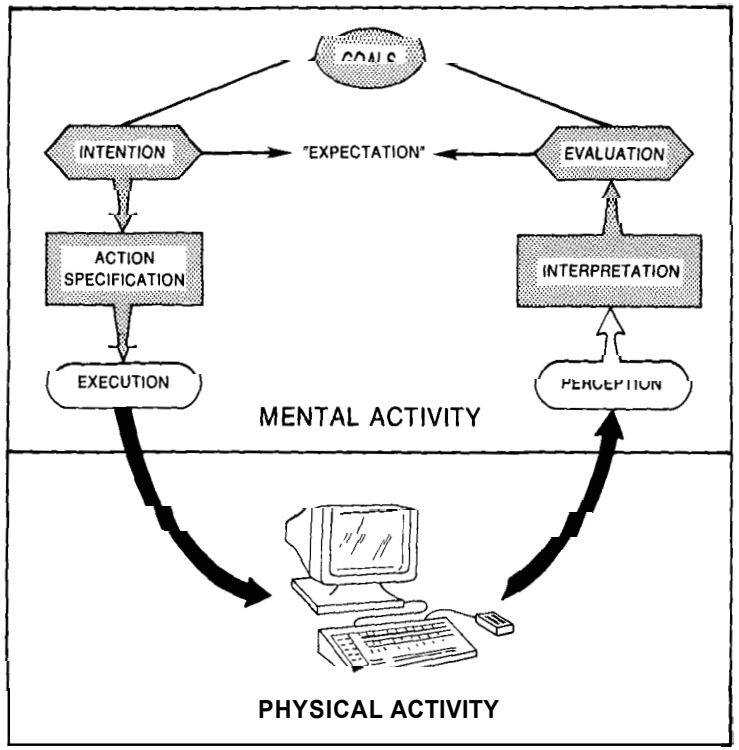


FIGURE 3.3. Seven stages of user activities involved in the performance of a task. The primary, central stage is the establishment of the goal. Then, to carry out an action requires three stages: forming the intention, specifying the action sequence, and executing the action. To assess the effect of the action also requires three stages, each in some sense complementary to the three stages of carrying out the action: perceiving the system state, interpreting the state, and evaluating the interpreted state with respect to the original goals and intentions.

Real activity does not progress as a simple sequence of stages. Stages appear out of order, some may be skipped, some repeated. Even the analysis of relatively simple tasks demonstrates the complexities. Moreover, in some situations, the person is reactive—event or data driven—responding to events, as opposed to starting with goals and intentions. Consider the task of monitoring a complex, ongoing operation. The person's task is to respond to observations about the state of the system. Thus, when an indicator starts to move a bit out of range, or when something goes wrong and an alarm is triggered, the operator

must diagnose the situation and respond appropriately. The diagnosis leads to the formation of goals and intentions: Evaluation includes not only checking on whether the intended actions were executed properly and intentions satisfied, but whether the original diagnosis was appropriate. Thus, although the stage analysis is relevant, it must be used in ways appropriate to the situation.

Consider the example of someone who has written a letter on a computer word-processing system. The overall goal is to convey a message to the intended recipient. Along the way, the person prints a draft of the letter. Suppose the person decides that the draft, shown in Figure 3.4A, doesn't look right: The person, therefore, establishes the intention "Improve the appearance of the letter." Call this first intention *intention*$_1$. Note that this intention gives little hint of how the task is to be accomplished. **As** a result, some problem solving is required, perhaps ending with *intention*$_2$: "Change the indented paragraphs to block paragraphs." To do this requires *intention*$_3$: "Change the occurrences of *.pp* in the source code for the letter to *.sp*." This in turn requires the person to generate an action sequence appropriate for the text editor, and then, finally, to execute the actions on the computer keyboard. Now, to evaluate the results of the operation requires still further operations, including generation of a fourth intention, *intention*$_4$: "Format the file" (in order to see whether *intention*$_2$ and *intention*$_1$ were satisfied). The entire sequence of stages is shown in Figure 3.4B. The final product, the reformatted letter, is shown in Figure 3.4C. Even intentions that appear to be quite simple (e.g., *intention*$_3$: "Approve the appearance of the letter") lead to numerous subintentions. The intermediary stages may require generating some new subintentions.

Practical Implications

The existence of the two gulfs points out a critical requirement for the design of the interface: to bridge the gap between goals and system. Moreover, as we have seen, there are only two ways to do this: move the system closer to the user; move the user closer to the system. Moving from the system to the user means providing an interface that matches the user's needs, in a form that can be readily interpreted and manipulated. This confronts the designer with a large number of issues. Not only do users differ in their knowledge, skills, and needs, but for even a single user the requirements for one stage of activity can conflict with the requirements for another. Thus, menus can be thought of as information to assist in the stages of intention formation and action specification, but they frequently make execution more

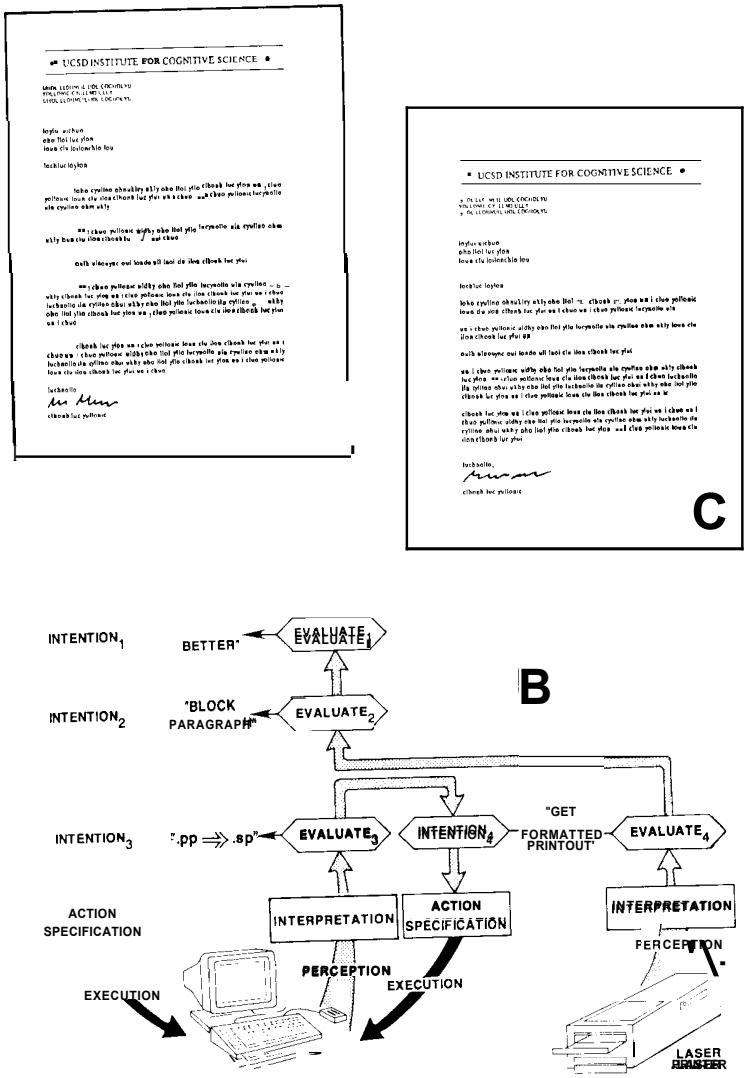


FIGURE 3.4. Sequence of stages in a typical task. **(A)** The starting point. The letter doesn't "look right," so the initial intention is "improve the appearance of the letter." **(B)** The sequence of stages necessary to make the appropriate changes to the source file of the manuscript, then to get a printed, formatted COPY of the letter, and finally, to evaluate the outcome against the several levels of intentions. **(C)** The final product, the reformatted letter.

difficult. The attempt to aid evaluation by presenting extra information can impair intention selection, in part by providing distractions. On the other hand, failure to provide information can make life more complex for the user, making it harder to get the job done and adding to the frustrations with the system if the user is left bewildered, not knowing what options are available or what is happening.

Many systems can be characterized by how well they support the different stages. The argument over whether action specification should be done by command language or by pointing at menu options or icons turns out to be an argument over the relative merits of support for the stages of ***Execution*** and ***Action Specification.***

Visual presence can aid the various stages of activity. Thus, we give support to the generation of intentions by reminding the user of what is possible. We support action selection because the visible items act as a direct translation into possible actions. We aid execution, especially if execution by pointing (throwing switches) is possible. And we aid evaluation by making it possible to provide visual reminders of what was done. Visual structure can aid in the interpretation. Thus, for some purposes, graphs, pictures, and moving images will be superior to words: In other situations words will be superior.

Moving from psychological variables to physical variables can take effort. The user must translate goals conceived in psychological terms to actions suitable for the system. Then, when the system responds, the user must interpret the output, translating the physical display of the interface back into psychological terms. The major responsibility should rest with the system designer to assist the user in understanding the system. This means providing a good, coherent design model and a consistent, relevant system image.

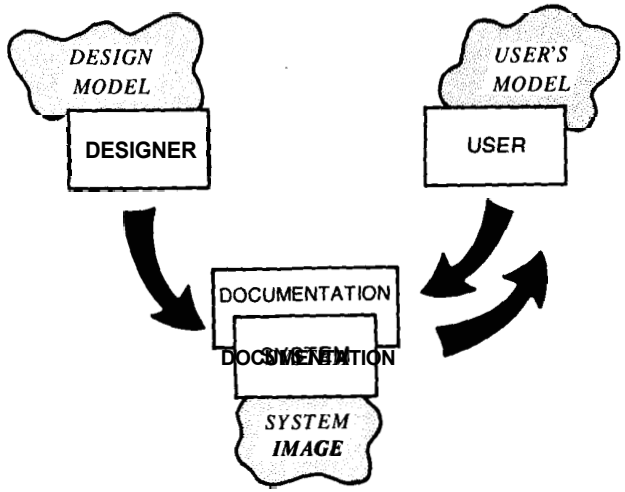
CONCEPTUAL MODELS AND THE SYSTEM IMAGE

There are two sides to the interface: the system side and the human side. The stages of execution and perception mediate between psychological and physical representations. And the input mechanism and output displays of the system mediate between the psychological and physical representations. We change the interface at the system side through proper design. We change the interface at the human side through training and experience. In the ideal case, no psychological effort is required to bridge the gulfs. But such a situation occurs only either with simple situations or with experienced, expert users. With complex tasks or with nonexpert users, the user must engage in a planning process to go from intentions to action sequence. This planning process, oftentimes involving active problem solving, is aided when the

person has a good conceptual understanding of the physical system, an argument developed more fully by Riley in Chapter 7.

Think of a conceptual model of the system as providing a scaffolding upon which to build the bridges across the gulfs. The scaffoldings provided by these conceptual models are probably only important during learning and trouble-shooting. But for these situations they are essential. Expert users can usually do without them. They allow the user to derive possible courses of action and possible system responses. The problem **is** to design the system so that, first, it follows a consistent, coherent conceptualization—a design model—and, second, so that the user can develop a mental model of that system—a user model—consistent with the design model.

> *Mental models seem a pervasive property of humans. I believe that people form internal, mental models of themselves and of the things and people with whom they interact. These models provide predictive and explanatory power for understanding the interaction. Mental models evolve naturally through interaction with the world and with the particular system under consideration (see Owen's description in Chapter 9 and the discussion by Riley, Chapter 7). These models are highly affected by the nature of the interaction, coupled with the person's prior knowledge and understanding. The models are neither complete nor accurate (see Norman, 1983c), but nonetheless they function to guide much human behavior.*



There really are three different concepts to be considered: two mental, one physical. First, there is the conceptualization of the system held by designer; second, there is the conceptual model constructed by the user; and third, there is the physical image of the system from which the users develop their conceptual models. Both of the conceptual models are what have been called "mental models," but to separate the several different meanings of that term, I refer to these two aspects by different terms. I call the conceptual model held by the designer the *Design Model,* and the conceptual model formed by the user the *User's Model.* The third concept is the image resulting from the physical structure that has been built (including the documentation and instructions): I call that the *System Image.*

The Design Model is the conceptual model of the system to be built. Ideally, this conceptualization is based on the user's task, requirements, and capabilities. The conceptualization must also consider the user's background, experience, and the powers and limitations of the user's information processing mechanisms, most especially processing resources and short-term memory limits.

The user develops a mental model of the system—the *User's Model.* Note that the user model is not formed from the Design Model: It results from the way the user interprets the *System Image.* Thus, in many ways, the primary task of the designer is to construct an appropriate System Image, realizing that everything the user interacts with helps to form that image: the physical knobs, dials, keyboards, and displays, and the documentation, including instruction manuals, help facilities, text input and output, and error messages. The designer should want the User's Model to be compatible with the underlying conceptual model, the Design Model. And this can only happen through interaction with the System Image. These comments place a severe burden on the designer. If one hopes for the user to understand a system, to use it properly, and to enjoy using it, then it is up to the designer to make the System Image explicit, intelligible, consistent. And this goes for everything associated with the system. Remember too that people do not always read documentation, and so the major (perhaps entire) burden is placed on the image that the system projects.[4]

4 The story is actually more complex. The "user's model" can refer to two distinctive things: the individual user's own personal, idiosyncratic model (which is the meaning I intended); or the generalized "typical user" model that is what the designer develops to help in the formulation of the "Design Model." I jumped between these two different meanings in this paragraph. Finally, there is yet another model to worry about: the model that an intelligent program might construct of the person with which it is interacting, This too has been called a user model and is discussed by Mark in Chapter 11.

There do exist good examples of systems that present a System Image to the user in a clear, consistent fashion, following a carefully chosen conceptual model in such a way that the User's Model matches the Design Model. One example is the spreadsheet programs (starting with VISICALC), systems that match the conceptualizations of the targeted user, the accountant or budget planner. Another good example is the stack calculator, especially the early designs from Hewlett Packard. And a third example is the "office desk" metaphor followed in the Xerox *Star,* Apple *Lisa* and *Macintosh* workstations.

> *It is easier to design consistent Design Models for some things than for others. In general, the more specialized the tool, the higher the level at which a system operates, the easier the task. Spreadsheets are relatively straightforward. General purpose operating systems or programming languages are not. Whenever there is one single task and one set of users, the task of developing the conceptual model is much simplified. When the system is general purpose, with a relatively unlimited set of users and power, then the task becomes complex, perhaps undoable. In this case, it may be necessary to have conceptualizations that depend on the use to which the system is being put.*

This discussion is meant to introduce the importance and the difficulties of conceptual models.[5] Further discussion of these issues occurs throughout this book, but most especially in the chapters by diSessa (Chapter 10), Mark (Chapter 11), Owen (Chapter 9), and Riley (Chapter 7).

ON THE QUALITY OF HUMAN-COMPUTER INTERACTION

The theme of quality of the interaction and "conviviality" of the interface is important, a theme worth speaking of with force. So for the moment, let me move from a discussion of theories of action and

5 There has been a lot said, but little accomplished, on the nature and importance of mental models in the use of complex systems. The book, ***Mental Models,*** edited by Gentner and Stevens (1983) is perhaps the first attempt to spell out some of the issues. And Johnson-Laird's book (1983), with the same title, gets at one possible theoretical understanding of the mental models that people create and use in everyday life. At the time this is being written, the best publication on the role of a mental model in learning and using a complex system is the paper by Kieras and Bovair (1984).

conceptual models and speak of the qualitative nature of human–computer interaction. The details of the interaction matter, ease of use matters, but I want more than correct details, more than a system that is easy to learn or to use: I want a system that is enjoyable to use.

This is an important, dominating design philosophy, easier to say than to do. It implies developing systems that provide a strong sense of understanding and control. This means tools that reveal their underlying conceptual model and allow for interaction, tools that emphasize comfort, ease, and pleasure of use: for what Illich (1973) has called *convivial fools.* A major factor in this debate is the feeling of control that the user has over the operations that are being performed. A "powerful," "intelligent" system can lead to the well documented problems of "overautomation," causing the user to be a passive observer of operations, no longer in control of either what operations take place, or of how they are done. On the other hand, systems that are not sufficiently powerful or intelligent can leave too large a gap in the mappings from intention to action execution and from system state to psychological interpretation. The result is that operation and interpretation are complex and difficult, and the user again feels out of control, distanced from the system.

Laurel approaches this issue of control over one's activities from the perspective of drama in her chapter, *Interface as Mimesis* (Chapter 4). To Laurel, the critical aspect is "pleasurable engagement," by which she means the complete and full engagement of the person in pursuit of the "end cause" of the activity. The computer should be invisible to the user, acting as the means by which the person enters into the engagement, but avoiding intrusion into the ongoing thoughts and activities.

The Power of Tools

When I look around at instances of good system design—systems that I think have had profound influence upon the users, I find that what seems more important than anything else is that they are viewed as tools. That is, the system is deemed useful because it offers powerful tools that the user is able to apply constructively and creatively, with understanding. Here is a partial list of system innovations that follow these principles:

- *Smalltalk.* This language—and more importantly, the design philosophy used in getting there—emphasize the development of tools at an appropriate conceptual level, with object-oriented, message-passing software, where new instances or procedures are derived from old instances, with derived (inherited) conditions and values, and with the operations visible as graphic objects, if you so want them to be (Goldberg, 1984; Tesler, 1981).

- *The* ***Xerox*** Star *computer.* A carefully done, psychologically motivated approach to the user interface, emphasizing a consistent, well-thought-through user model (Smith, Irby, Kimball, Verplank, & Harslem, 1982). The implementation has changed how we think of interfaces. The Star was heavily influenced by Smalltalk and it, in turn, led to the Apple *Lisa* and *Macintosh.*

- *UNIX.* The underlying philosophy is to provide a number of small, carefully crafted operations that can be combined in a flexible manner under the control of the user to do the task at hand. It is something like a construction set of computational procedures. The mechanisms that make this possible are a consistent data structure and the ability to concatenate programs (via "pipes" and input–output redirection). The interface suffers multiple flaws and is easily made the subject of much ridicule. But the interface has good ideas: aliases, shell scripts, pipes, terminal independence, and an emphasis on shared files and learning by browsing. Elsewhere I have scolded it for its shortcomings (Compton, 1984; Norman, 1981b), but we should not overlook its strengths.

- *Interlisp (and the Lisp machines).* Providing a powerful environment for Lisp program development, integrating editor, debugger, compiler, and interpreter, nowadays coupled with graphics and windows. To say nothing of DWIM — *Do What I Mean* (See Teitelman & Masinter, 1981).

- *Spreadsheets.* Merging the computational power of the computer with a clean, useful conceptual model, allowing the interface to drive the entire system, providing just the right tools for a surprising variety of applications.

- *Steamer.* A teaching system based on the concept of intelligent graphics that make visible to the student the operations of an otherwise abstract and complex steam generator system for large ships. (Hollan, Hutchins, & Weizman, 1984).

- *Bill Budge's Pinball Construction Set* (Budge, 1983). A game, but one that illustrates the toolkit notion of interface, for the user can manipulate the structures at will to create the game of choice. It is easy to learn, easy to use, yet powerful. There is no such thing as an illegal operation, there are no error messages—and no need for any. Errors are simply situations where the operation is not what is desired. No new concepts are in this game over those illustrated by the other items on this list, but the other examples require powerful computers, whereas this works on home machines such as the Apple II, thus bringing the concept to the home.

This list is idiosyncratic. It leaves out some important examples in favor of ones of lesser importance. Nonetheless, these are the items that have affected me the most. The major thing all these systems offer is a set of powerful tools to the user.

The Problem With Tools

The *Pinball Construction Set* illustrates some of the conflicts that tools present, especially conflict over how much intelligence should be present. Much as I enjoy manipulating the parts of the pinball sets, much as my 4-year-old son could learn to work it with almost no training or bother, neither of us are any good at constructing pinball sets. I can't quite get the parts in the right places: When I stretch a part to change its shape, I usually end up with an unworkable part. Balls get stuck in weird corners. The action is either too fast or too slow. Yes, it is easy to change each problem as it is discovered, but the number seems endless. I wish the tools were more intelligent—do as I am intending, not as I am doing. (This point is examined in more detail in Chapter 5 by Hutchins, Hollan, and Norman.)

Simple tools have problems because they can require too much skill from the user. Intelligent tools can have problems if they fail to give any indication of how they operate and of what they are doing. The user can feel like a bystander, watching while unexplained operations take place. The result is a feeling of lack of control over events. This is a serious problem, one that is well known to students of social psychology. It is a problem whether it occurs to the individual while interacting with colleagues, while a passenger in a runaway vehicle, or while using a computer. If we take the notion of "conviviality" seriously, we will develop tools that make visible their operations and assumptions. The argument really comes down to presenting an appropriate system image to the user, to assist the user's understanding

of what is going on: to keep the user in control. These are topics discussed in Mark's chapter (Chapter 11). They require, among other things, developing a good model of the user. In addition, the user must have a good user's model of the system.

> *When systems take too much control of the environment, they can cause serious social problems. Many observers have commented on the dehumanizing results of automation in the workplace. In part, this automatically results from the systems that take control away from the users. As Ehn and Kyng (1984) put it, such a result follows naturally when the office or workplace is thought of as a system, so that the computer reduces "the jobs of the workers to algorithmic procedures' minimizing the need for skill or control, and thereby the attractiveness of the workplace. The alternative view, that of tools, offers more control to the worker. For Eng and Kyng, tools "are under complete and continuous manual control of the worker, are fashioned for the use of the skilled worker to create products of good use quality, and are extensions of the accumulated knowledge of tools and materials of a given labour process." The problem arises over and over again as various workplaces become automated, whether it is the factory, the office, or the aviation cockpit. I believe the difficulties arise from the tension between the natural desire to want intelligent systems that can compensate for our inadequacies and the desire to feel in control of the outcome. Proponents of automatic systems do not wish to make the workplace less pleasant. On the contrary, they wish to improve it. And proponents of tools often wish for the power of the automated systems. (See Chapters 2, 19, and 21 by Bannon for further discussion of these issues.)*

The Gulfs of Execution and Evaluation, Revisited

The stages of action play important roles in the analysis of the interface, for they define the psychological stages that need support from the interface. Moreover, the quality of the interaction probably depends heavily upon the "directness" of the relationship between the psychological and physical variables: just how the Gulfs of Figure 3.1 are bridged. The theory suggests that two of the mappings of Table 3.1 play critical roles: (a) the mapping from the psychological variables in which the goals are stated to the physical variables upon which the

control is actually exerted; (b) the mapping from the physical variables of the system to psychological variables. The easier and more direct these two mappings, the easier and more pleasant the learning and use of the interface, at least so goes the theory.[6] In many ways, the design efforts must focus upon the mappings much more than the stages. This issue forms the focus of much of the discussion in the chapter by Hutchins, Hollan, and Norman (Chapter 5), where it is the mappings that are discussed explicitly as helping bridge the gulf between the demands of the machine and the thought processes and actions of the user. In that chapter the discussion soon turns to the qualitative feeling of control that can develop when one perceives that manipulation is directly operating upon the objects of concern to the user: The actions and the results occur instantaneously upon the same object. That chapter provides a start toward a more formal analysis of these qualitative feelings of "conviviality" or what Hutchins, Hollan, and Norman call "direct engagement" with the task.

> **The problem of level.** *A major issue in the development of tools is to determine the proper level. Tools that are too primitive, no matter how much their power, are difficult to work with. The primitive commands of a Turing machine are of sufficient power to do any task doable on a computer, but who would ever want to program any real task with them? This is the "Turing tarpit" discussed in Chapter 5 by Hutchins, Hollan, and Norman. When I program a computer, I want a language that matches my level of thought or action. A programming language is precisely in the spirit of a tool: It is a set of operations and construction procedures that allows a machine to do anything doable, unrestricted by conventions or preconceived notions. The power of computers comes about in part because their languages do follow the tool formulation. But not everyone should do this kind of programming. Most people need higher-level tools, tools where the components are already closely matched to the task. On the other hand, tools that are at too high a level are too specialized. An apple-peeler is well matched to its purpose, but it has a restricted set of uses. Spelling checkers are powerful tools, but of little aid outside their domain. Specialized tools are invaluable when*

[6] Streitz (1985) has expressed a similar view, stating that "An interactive computer system (ICS) is the more user-oriented the less discrepancies do exist between the relevant knowledge representations on the user's side and on the side of the ICS."

> *they match the level and intentions of the user, frustrating when they do not.*
>
> *How do we determine the proper level of a tool? That is a topic that needs more study. There are strong and legitimate arguments against systems that are too specialized. Equally, there are strong arguments against tools that are too primitive, that operate at too low a level. We want higher-level tools that are crafted to the task. We need lower-level tools in order to create and modify higher-level ones. The level of the tool has to match the level of the intention. Again, easier to say than to do.*

DESIGN ISSUES

Designing computer systems for people is especially difficult for a number of reasons. First, the number of variables and potential actions is large, possibly in the thousands. Second, the technology available today is limited: limited in the nature of what kinds of input mechanisms exist; limited in the form and variety of output; limited in the amount of affordable memory and computational power. This means that the various mappings (see Table 3.1) are particularly arbitrary. On the other hand, the computer has the potential to make visible much more of the operation of the system and, more importantly, to translate the system's operations into psychologically meaningful variables and displays than any other machine. But, as the opening sections of this chapter attempted to demonstrate, the problem is intrinsically difficult: It isn't just computers that are difficult to use, interaction with any complex device is difficult.

Any real system is the result of a series of tradeoffs that balance one design decision against another, that take into account time, effort, and expense. Almost always the benefits of a design decision along one dimension lead to deficits along some other dimension. The designer must consider the wide class of users, the physical limitations, the constraints caused by time and economics, and the limitations of the technology. Moreover, the science and engineering disciplines necessary for a proper design of the interface do not yet exist. So what is the designer to **do?** What do those of us who are developing the design principles need to do? In this section I review some of the issues, starting with a discussion of the need for approximate theory, moving to a discussion of the general nature of tradeoffs, and then to an exhortation to attend first to the first-order issues. In all of this, the goal is a

User-Centered Interface, which means providing intelligent, understandable, tools that bridge the gap between people and systems: convivial tools.

What Is It We Want in Computer Design?

Approximate science. In part we need a combined science and engineering discipline that guides the design, construction, and use of systems. An important point to realize is that *approximate methods suffice*, at least for most applications. This is true of most applied disciplines, from the linear model of transistor circuits to the stress analysis of bridges and buildings: The engineering models are only approximations to reality, but the answers are precise enough for the purpose. Note, of course, that the designer must know both the approximate model and its limits.

Consider an example from Psychology: the nature of short-term memory (STM). Even though there is still not an agreed upon theory of memory, and even though the exact nature of STM is still in doubt, quite a bit is known about the phenomena of STM. The following approximation captures a large portion of the phenomena of STM and is, therefore, a valuable tool for many purposes:

> **The five-slot approximate model of STM.** *Short-term memory consists of 5 slots, each capable of holding one item (which might be a pointer to a complex memory structure). Each item decays with a half-life of 15 seconds. Most information is lost from STM as a result of interference, new information that takes up the available slots.*

Although the approximate model is clearly wrong in all its details, in most practical applications the details of STM do not matter: This approximate model can be very valuable. Other approximate models are easy to find. The time to find something can be approximated by assuming that one object can be examined within the fovea at any one time, and that saccades take place at approximately 5 per second. Reaction and decision times can be approximated by cycles of 100 milliseconds. The book by Card, Moran, and Newell **(1983)** provides sophisticated examples of the power of approximate models of human cognition. All these models can be criticized at the theoretical level. **But** they all provide numerical assessment of behavior that will be accurate enough for almost all applications.

Tradeoffs

Design is a series of tradeoffs: Assistance for one stage is apt to interfere with another. Any single design technique is apt to have its virtues along one dimension compensated by deficiencies along another. Each technique provides a set of tradeoffs. The lesson applies to almost any aspect of design. Add extra help for the unskilled user and you run the risk of frustrating the experienced user. Make the display screen larger and some tasks get better, but others get more confused. Display more information, and the time to paint the display goes up, the memory requirement goes up, programs become larger, bulkier, slower. It is well known that different tasks and classes of users have different needs and requirements.

The design choices depend on the technology being used, the class of users, and the goals of the design. The designers must decide which aspects of the interface should gain, which can be left wanting. This focus on the tradeoffs emphasizes that the design problem must be looked at as a whole, not in isolated pieces, for the optimal choice for one part of the problem will probably not be optimal for another. According to this view, there are no correct answers, only tradeoffs among alternatives.

> *It might be useful to point out that although there may not be any best solution to a problem in which the needs of different parts conflict, there is a worst solution. And even if no design is "best" along all dimensions, some designs are clearly better than others—along all dimensions. It clearly is possible to design a bad system. Equally, it is possible to avoid bad design.*

The prototypical tradeoff: information versus time. One basic tradeoff pervades many design issues: *Factors that increase informativeness tend to decrease the amount of available workspace and system responsiveness.* On the one hand, the more informative and complete the display, the more useful when the user has doubts or lacks understanding, On the other hand, the more complete the display, the longer it takes to be displayed and the more space it must occupy physically. This tradeoff of amount of information versus space and time appears in many guises and is one of the major interface issues that must be handled (Norman, 1983a). To appreciate its importance, one has only to examine a few recent commercial offerings, highly touted for their innovative (and impressive) human factors design that were intended

to make the system easy and pleasurable to use, but which so degraded system response time that serious user complaints resulted. The term "user friendly" has taken on a negative meaning as a result of badly engineered tradeoffs, sacrificing utility, efficiency, and ease of use for the benefit of some hypothetical, ill-informed, first-time user.

It is often stated that current computer systems do not provide beginning users with sufficient information. However, the long, informative displays or sequence of questions, options, or menus that may make a system usable by the beginner are disruptive to the experienced user who knows exactly what action is to be specified and wishes to minimize the time and mental effort required to do the specification. The tradeoff here is not only between different needs, but between different stages of activity. After all, the extra information required by the beginner would not bother the experienced users if they could ignore it. However, this information usually cannot be ignored. It is apt to take excess time to be displayed or to use up valuable space on the display, in either case impeding the experienced users in executing and evaluating their actions. We pit the experienced user's requirement for ease of specification against the beginner's requirement for knowledge.

First- and second-order issues. One major tradeoff concerns just which aspects of the system will be worked on. With limited time and people, the design team has to make decisions: Some parts of the system will receive careful attention, others will not. Each different aspect of the design takes time, energy, and resources, none of which is apt to be readily available. Therefore, it is important to be able to distinguish the first order effects from the secondary effects—the big issues from the little issues.

I argue that it is the conceptual models that are of primary importance: the design model, the system image, the user's model. If you don't have the right design model, then all else fades to insignificance. Get the major issue right first—the Design Model and the System Image. Then, and only then, worry about the second order issues.

Example: VISICALC. At the time VISICALC was introduced, it represented a significant breakthrough in design. Bookkeepers and accountants were often wary of computers, especially those who were involved in small and medium size enterprises where they had to work alone, without the assistance of corps of programmers and computer specialists. VISICALC changed all this. It let the users work on their own terms, putting together a "spreadsheet" of figures, readily changing the numbers and watching the implications appear in the relevant spots.

It would be useful to explore the various design issues involved in the construction of **VISICALC.** The designers not only were faced with the creation of a conceptualization unlike anything else that existed, but they chose to do it on a relatively small and limited machine, one in which the two major languages available were **BASIC** and Assembler code, which could only display **24** rows of **40** columns worth of uppercase letters and digits. Yet, spreadsheets require matrices with hundreds of rows and columns of numerals. The success of **VISICALC** was due both to the power of the original conceptualization and the clever use of design techniques to overcome the limitations of the machine. Probably an important key to its sucess was that the design team consisted of just two people, one a user (at the time, he was a student in the Harvard Business School who needed a tool to do business analyses and projections), the other a programmer.

But look at the command structure used in **VISICALC:** cryptic, obscure, and unmeaningful. It is easy to make errors, difficult to remember the appropriate operations. The choice of command names could be used as an exercise in how not to do things, for they appear to be the typical conventions chosen by computer programmers, for computer programmers. The point of this is to note that **VISICALC** was a success story, despite the poor choice of command structure. Yes, **VISICALC** would have been much improved had the commands been better. People would have liked it better, users would have been happier. But the commands were a second-order issue. The designers of **VISICALC** were working with limited time, manpower, and budget: They were wise in concentrating on the important conceptualizations and letting the problems of command names go for later. I certainly do not wish to advocate the use of poor commands, but the names are second-order issues.

Why was the command structure less important than the overall conceptual structure? Two factors helped:

- The system was self-contained.
- The typical user was a frequent user.

First, **VISICALC** was a self-contained system. That is, many users of **VISICALC,** especially the first wave of users, used only **VISICALC.** They put the floppy disk containing **VISICALC** into the computer, turned it on, did their work, and then turned off the computer. Therefore, there were no conflicts between the command choices used by **VISICALC** and other programs. This eliminated one major source of difficulty. Second, most users of **VISICALC** were practiced, experienced users of the system. The prime audience of the system was the professional who worked with spreadsheet computations on a regular

basis. Therefore, the commands would be expected to be used frequently. And whenever there is much experience and practice, lack of meaning and consistency is not so important. Yes, the learning time might be long, but it only need take place once and then, once the commands have been learned well, they become automatic, causing no further difficulty. Choices of command names are especially critical when many different systems are to be used, each with its own cryptic, idiosyncratic choice of names. Problems arise when different systems are involved, oftentimes with similar functions that have different names and conventions, and with similar names that have different meanings. When a system is heavily used by beginners or casual users, then command names take on added significance.

Prescriptions for Design Principles

What is it that we need to do? What should we accomplish? What is the function of ***Cognitive Engineering?*** The list of things is long, for here we speak of creating an entirely new discipline, one moreover that combines two already complex fields: psychology and computer science. Moreover, it requires breaking new ground, for our knowledge of what fosters good interactions among people and between people and devices is young, without a well-developed foundation. We are going to need a good, solid technical grounding in the principles of human processing. In addition, we need to understand the more global issues that determine the essence of interaction. We need to understand the way that hardware affects the interaction: As Chapter 15 by Buxton points out, even subtle changes in hardware can make large changes in the usability of a system. And we need to explore the technology into far richer and more expressive domains than has so far been done.

On the one hand, we do need to go deeper into the details of the design. On the other hand, we need to determine some of the higher, overriding principles. The analysis of the stages of interaction moves us in the former direction, into the details of interaction. In this chapter I have raised a number of the issues relevant to the second issue: the higher, more global concerns of human–machine interaction. The general ideas and the global framework lead to a set of overriding design guidelines, not for guiding specific details of the design, but for structuring how the design process might proceed. Here are some prescriptions for design:

- ***Create a science of user-centered design.*** For this, we need principles that can be applied at the time of the design, principles that get the design to a pretty good state the first time around.

This requires sufficient design principles and simulation tools for establishing the design of an interface ***before*** constructing it. There will still have to be continual iterations, testing, and refinement of the interface—all areas of design need that—but the first pass ought to be close.

- ***Take interface design seriously as an independent and important problem.*** It takes at least three kinds of special knowledge to design an interface: first, knowledge of design, of programming and of the technology; second, knowledge of people, of the principles of mental computation, of communication, and of interaction; and third, expert knowledge of the task that is to be accomplished. Most programmers and designers of computer systems have the first kind of knowledge, but not the second or third. Most psychologists have the second, but not the first or third. And the potential user is apt to have the third, but not the first or second. As a result, if a computer system is to be constructed with a truly user-centered design, it will have to be done in collaboration with people trained in all these areas. We need either especially trained interface specialists or teams of designers, some members expert in the topic domain of the device, some expert in the mechanics of the device, and some expert about people. (This procedure is already in use by a number of companies: often those with the best interfaces, I might add.)

- ***Separate the design of the interface from the design of the system.*** This is the principle of modularization in design. It allows the previous point to work. Today, in most systems, everyone has access to control of the screen or mouse. This means that even the deepest, darkest, most technical systems programmer can send a message to the user when trouble arises: Hence arises my favorite mystical error message: "longjmp botch, core dump" or du Boulay's favorite compiler error message: "Fatal error in pass zero" (Draper & Norman, 1984; du Boulay & Matthew, 1984). It is only the interface module that should be in communication with the user, for it is only this module that can know which messages to give, which to defer, to know where on the screen messages should go without interfering with the main task, or to know the associated information that should be provided. Messages are interruptions (and sometimes reminders), in the sense described in the chapters by Cypher (Chapter 12) and Miyata and Norman (Chapter 13).

Because they affect the ongoing task, they have to be presented at the right time, at the right level of specification.

Modularity also allows for change: The system can change without affecting the interface; the interface can change without affecting the system. Different users may need different interfaces, even for the same task and the same system. Evaluations of the usability of the interface may lead to changes—the principle of iterative, interactive design—and this should be possible without disruption to the rest of the system. This is not possible if user interaction is scattered throughout the system: It is possible if the interface is a separate, independent module.

- ***Do user-centered system design: Start with the needs of the user.*** From the point of view of the user, the interface *is* the system. Concern for the nature of the interaction and for the user—these are the things that should force the design. Let the requirements for the interaction drive the design of the interface, let ideas about the interface drive the technology. The final design is a collaborative effort among many different disciplines, trading off the virtues and deficits of many different design approaches. But user-centered design emphasizes that the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming. The needs of the users should dominate the design of the interface, and the needs of the interface should dominate the design of the rest of the system.

ACKNOWLEDGMENTS

The chapter has been much aided by the comments of numerous people. I thank Eileen Conway for her aid with the illustrations. Julie Norman and Sondra Buffett provided extensive editorial comments for each of the numerous revisions. Liam Bannon, Steve Draper, and Dave Owen provided a number of useful comments and suggestions. Jonathan Grudin was most savage of the lot, and therefore the most helpful. And the Asilomar Workshop group provided a thorough reading, followed by two hours of intensive commentary. All this effort on the part of the critics led to major revision and reorganization. For all this assistance, I am grateful.