ALLEN AVNER
Senior Specialist in Automated Education
Director of Evaluation
Computer-based Education Research Laboratory
University of Illinois at Urbana-Champaign

H. GEORGE FRIEDMAN, JR.
Associate Professor of Computer Science
University of Illinois at Urbana-Champaign

Interacting with Computer Users: Design Considerations

The design of computer terminals which communicate with naïve users in a humane yet effective manner involves problems that are common to many applications. Interactive computers are being used as everyday tools in settings that range from airlines to zoos. The problems met by the growing use of computer terminals in libraries are seldom unique to that setting. In fact, the experience of the authors over the past two decades suggests the existence of a set of design problems that turn up whenever interactive computer terminals are used, whatever the setting. These hindrances emanate from inappropriate dependence on a few simplifying assumptions that make design easier at the cost of lowered effectiveness. This paper outlines six of these fallacious assumptions, describes the reasons for their beguiling attractiveness, and suggests alternative views that should lead to better design.

Human-Machine System Design

Human-machine interactions may be considered to consist of four major elements: task, procedure, human, and machine. The good system designer does not assume that any of these elements is a static, unchangeable factor that can be ignored. A thorough analysis may even reveal alternative approaches which eliminate the need for a special design.

Task

The task is the problem that is to be solved. A common error is the failure to understand that past views of the problems may have been limited by what was possible with tools and procedures then available.

New tools may make it possible to solve a larger problem. Where this is the case, the task should be defined to include factors not addressed by former approaches. An example common in computerization is that a computer brought in to simplify paperwork also turns out to be able to solve a part of the management process that the paperwork supported. Thus, a computer brought in to automate the production of book purchase orders would also be able to automate many of the standard administrative decisions made in following up on overdue deliveries. However, this added capability is likely to be included in the design of the system only if the designer is aware of the total task.

Procedure

Procedures are the methods used to complete a task. A common error is the confusion of procedures with tasks. A procedure (such as filling out a charge slip for a book by hand) comes to be seen as a required part of an operation, rather than as simply one of several means of performing the actual task (maintaining a record of changes in responsibility for a book). This particular confusion often results in technological misapplications that meet task needs by the simple, but usually inefficient, expedient of mimicking old procedures.

Another form of confusion of task and procedure results in the endowment of a procedure with almost magical powers. Thus, "computerization" may be cited as the reason for success of a new approach. That success is then used as the reason for blindly adopting computers in other situations without taking the trouble to determine what alternatives might be available. One result of such blind adoption of computers is the growing number of cases where an administrator "computerizes" an operation, shows great savings in time and money, and is promoted. He is then replaced by a new administrator who is miraculously able to eliminate the computer without losing the advantages of "computerization"! If the first administrator had taken the time to examine the actual task and the possible alternatives, he would have observed that all that was really needed was a restructuring of the task. The apparent gains derived from computerization in such cases really result from the task restructuring that accompanies the unnecessary addition of a computer. The computer can be dropped from such an implementation with minimal effect on working efficiency and substantial savings in cost. Needless to say, cost savings would have been even greater if the computer had never entered the scene.

Human

Humans appear in many roles in human-machine systems. They may help the machine carry out procedures, or they may be clients served by the

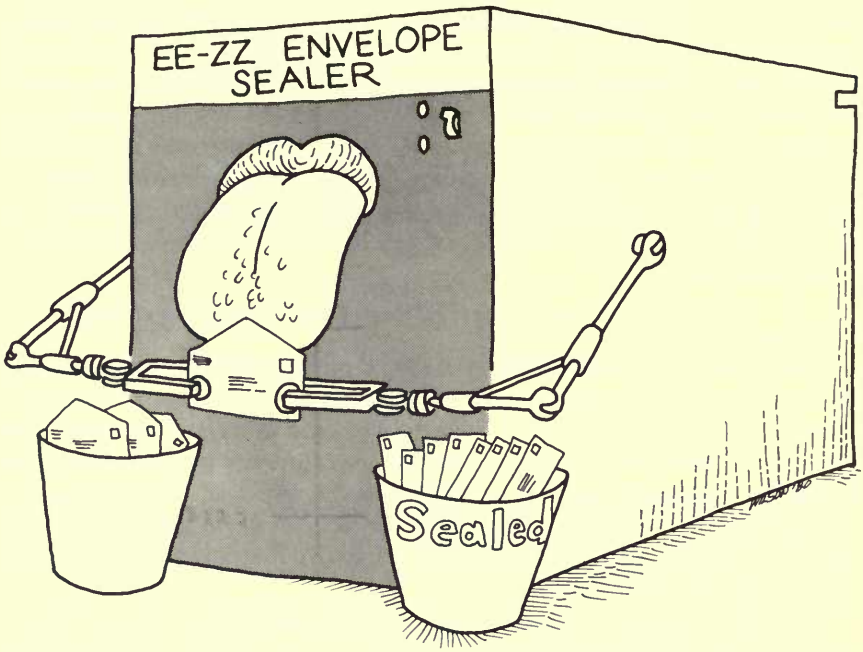

Figure 1. Beware of technological innovations that simply mimic old procedures.

Illustrations for this paper were drawn by Wayne Wilson, Computer-based Education Research Laboratory, University of Illinois at Urbana-Champaign.

system. The major error lies in ignoring the human element. A designer may assume that since he or she is human, any system design will automatically include all needed human factors. That is not so. Humans vary enormously in training, motivation and ability. Not only do they vary individually, but they vary with time. A system designed for naïve users may not be efficient for experienced users. Users who begin as naïve users will seldom remain that way with time. A system that is comfortable for brief human use can be an unbearable burden when used continuously for long hours. Good design demands a clear view of the nature of the humans who will interact with the system and the nature of that interaction.

Machine

The word *machine* is used here because we happen to be discussing computers. A more accurate word would be *tool*. The machine or tool is a technological aid to application of the procedure. A pencil or an instruc-

tional technique is just as valid a form of technology as a computer. It is a mistake to assume that some complex form of technology is needed for every task. As was noted above, careful restructuring of a work situation can result in substantial improvements in productivity without any need for a computer or other expensive technology.

Modularization

Once the major components of a human/machine process have been identified, good design practice dictates that the resulting system be further broken down into functional modules that cut across these major components. Each functional module performs a single distinct function in the solution of the overall task. A particular module is defined by the portion of the total task that it covers, the procedures needed to address that subtask, and the human and/or machine carrying out those procedures. Each module has well-defined inputs and outputs, and most modules interact only with other modules.

Modularization is not done simply in response to an innate drive of system designers to categorize things. Modularization allows concurrent development of the many parts of a complex system by different groups of designers operating in relative independence, thus greatly reducing the length of time between initial planning and putting a system into operation. Modularization also has advantages in the completed product.

Both hardware (the terminals, computers and other devices associated with the system) and software (the computer programs which guide the hardware and interpret interactions between hardware and humans) can be modularized. Modularized hardware is easier to maintain, since modules that serve only a single, well-defined function are easier to isolate should they malfunction. Modules are also amenable to quick, inexpensive repair by substitution. Properly designed modular hardware is also more easily altered or expanded to meet changing needs of a given installation. For example, needs for additional terminals or increased information storage capability can be met simply by adding the needed equipment and the control modules required to interface it with the original system. The same expansion in a nonmodularized system might require major redesigning of both hardware and software. Modularized software has similar advantages in identification and repair of problems and in modification of existing installations.

Unfortunately, the advantages of modularization can cause a designer to downgrade the importance of other design considerations. For example, modularization is easiest when the task structure is relatively simple and when there are few interactions between tasks or procedures of different

types. In seeking such simplicity, a good system designer tries to eliminate extraneous elements from the task that the system is to perform. Under time pressure, however, such commendable parsimony can lead to oversimplification. Oversimplification results either from failures of commission (misinterpreting a user's description of the task) or failures of omission (failing to verify that an interpretation of the task actually leads to an acceptable final result). The fault in failures of omission is not always with the system designer alone. A user who is not familiar with the computer's slavishly literal interpretation of directives may fail to specify the crucial decisions that are often made by a human faced with ambiguous information. A human is able to make commonsense interpretations that may result in the job's completion despite less than ideal information. A computer programmed with an oversimplified procedure for handling the same ambiguous data may merrily grind out stacks of absolute rubbish. Eventually such failures will come to light, of course, but it is far more efficient to identify them at the time the system design is being specified. In the early design stages, no amount of experience in computer system design can replace the knowledgeable guidance of a person who has actually carried out the original task under "real-life" conditions.

With this general background, let us examine six of the most common fallacies that intrude on the design of interactive computer terminal systems. We hope that once you are aware of these pitfalls to good design, you will be better able to guide design or selection of an interactive terminal system that will meet the special needs of your application.

The Fallacy of Subsystem Independence

The person following this erroneous design principle assumes that any component of a system can be designed effectively without any knowledge of the rest of the system. It is both convenient and useful to handle design of a system by breaking the major system into component modules. This does not mean, however, that the final system is intended to function as a set of independent modules.

Problems of the "Fallacy of Subsystem Independence" show themselves most frequently in hardware interactions. At the lowest level, the user might encounter massive delays in accessing or storing information at a terminal that is mismatched to a communications channel or storage device. The fact that two such components can be made to communicate with each other by means of intermediate software or hardware does not necessarily mean that the interaction will be efficient.

At a more complex (and, unfortunately, more often observed) level, a system might perform a variety of functions quite well when the system is

supporting only one user. However, a system which has not been designed as an integrated whole may show severely degraded performance when asked to support different operations simultaneously by several users at different interactive terminals. It may even degrade severely simply as a result of user loads above some moderate level.

The Fallacy of System Function

The motto of the believer in this fallacy is "If it works, the design is okay." No designer intentionally designs a system that is difficult to use, performs inefficiently, or assumes an inordinate amount of skill on the part of the user. Nevertheless, under pressure to produce a functioning system while facing the ever present time deadlines and funding limitations that mark reality, designers are too frequently willing to accept almost anything that actually gets the intended task done. They may have started out with far grander intentions and an abiding desire to produce a system that would be both a joy to use and a paragon of efficiency. But in the cold, hard dawn of reality (and corporate solvency), they may have been willing to compromise with something that met the minimum specifications of the contract.

Given the tendency of humans to compromise when under pressure, it is wise to make sure that minimal contractual specifications will actually provide acceptable levels of performance in the finished system. To insure that minimal specifications are adequate, one must go beyond simple statements of input information and output products. One must identify important conditional factors such as speed and ease of operation, and specify the operating condition under which these performance levels are expected.

A system must be expected to perform differently under different usage loads. In recognition of this fact of life, the levels of performance required under a "normal" load and under the most severe load anticipated should both be specified. We must always remember that even if a system "works" (i.e., produces the desired results under ideal conditions), it will not necessarily be acceptable to users (i.e., produce the desired results in a real life setting).

The Fallacy of Human Perspicacity

This fallacy is committed by most persons involved with the design of interactive systems. The assumption that all humans will think exactly the way *you* think (and that they will automatically understand your intent at each step of an interaction) is woefully common. The more involved a

designer becomes with the mechanics of getting a system to "work," the more he or she grows accustomed to the idiosyncratic manner in which it happens to operate at that moment. After a while, the designer forgets that everyone will not come to the system with a full understanding of the intent behind each human-machine interaction.

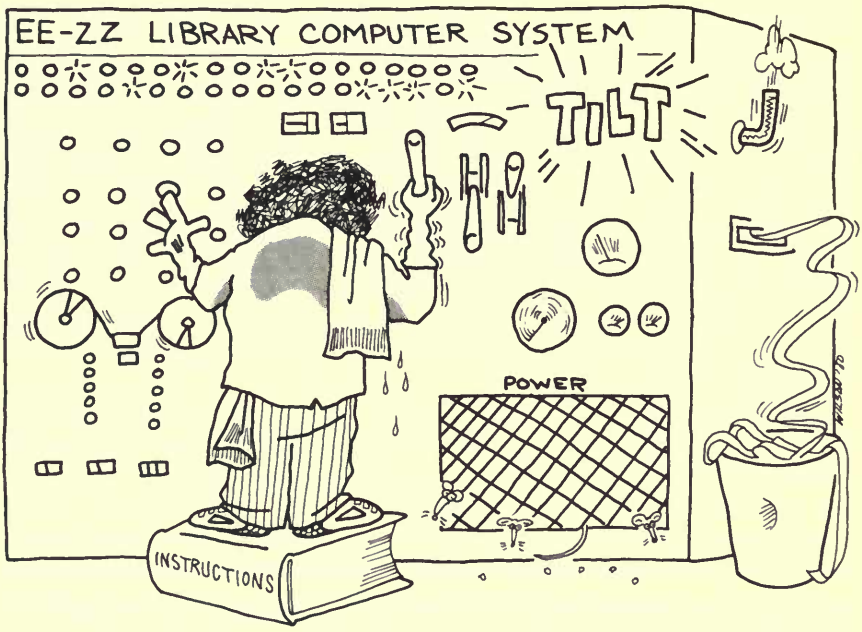

Figure 2. A foot-thick instruction manual is no substitute for good human-machine design.

The day of reckoning arrives with the first use by people who were not involved with the original design. If the designers are wise, such people will be brought in well before the design of human-machine interactions has been frozen. Careful study of the problems encountered by naïve users will greatly aid in reducing potential errors and in increasing the ease of interaction. If the designers are *not* wise, they will delay exposure of the system to realistic field testing until final delivery of the system. Systems designed under this "blind" approach are most notable for their literary contributions—a frantic, last-minute effort to compensate for poor human engineering by provision of a flood of instruction manuals. In general, the

less often a user interacts with a computer terminal, the less voluminous the printed instruction manuals should be. A well-designed interactive computer system provides complete interactive prompting for the new or infrequent user. Ideally, one should be able to start a new user by simply saying, "Follow the directions on that screen." Assuming reasonably literate users, anything less than this should be taken as a possible sign of limitations either in hardware capabilities or, more likely, in software design.

The Fallacy of Human Memory

A fallacy which is particularly prevalent in the design of information and instruction displays for interactive terminal systems is the assumption that humans can remember every detail of information encountered several minutes or even seconds before. This fallacy is actually a special case of the "Fallacy of Human Perspicacity" and is perpetrated for the same reason. After many days of working with the structure of a system, one forgets that someone seeing the system for the first time will, for example, actually be using instructions as sources of new information and not simply as mile markers on a familiar path.

In the brief exchanges characteristic of use of interactive computer terminals, humans depend mostly on short-term memory. This is the same type of memory that we use for tasks such as remembering a telephone number from the time we look it up in a directory to the time we dial it on the telephone. Short-term memory normally has a very limited capacity (about three to four simple items or groups of items) and is easily overwritten by new information.[1] It is not reasonable to expect people to take the time to memorize directions on a display which they have little intention of using frequently. Nor, given the limitations of short-term memory, is it reasonable to expect a human to remember an item of information from one display and combine it with information from another display. While this is a task that *can* be done, it is a task that forces a human to do something a computer can do far better.

Well-designed interactions provide directions appropriate to the needs of the user at the moment they are needed. Well-designed interactions also keep track of information acquired by the user (e.g., in a search procedure) and permit easy recovery of that information. For example, after completing a complex search operation, the user should be able to make a minor change in specifications and start a new search without having to retype the full set of specifications.

The Fallacy of Human Patience

Time is probably the most frequently overlooked incidental factor in system performance. Designs that ignore the effect of delays in system response on user acceptance or performance are implicitly assuming that such effects do not exist. Rest assured, they do. Short delays—for example, when a user types a letter and nothing appears immediately on the display—can convince the user that the system has not seen an input. Delays as short as a quarter-second can lead users to make repeated inputs. The repeated input leads in turn to errors (a double letter where a single letter was intended) or to wasted resources (two requests for recovery of data when only one was desired). Longer delays can lead to user frustration.

Most frustrating of all are delays of random duration. One moment the user receives almost instantaneous service and the next moment the user must wait for what seems to be an eternity. Variable delays are generally the result of variations in load. Instant response is available when a single user is present, but delays become noticeable as more users attempt simultaneous use of system resources. In a well-designed system, loading effects should not be perceptible for rapid sequential operations (such as typing the separate letters of a name) and should be minimal for major operations (such as the delay between initiating a title search and first seeing the results of that search). Response time for rapid sequential operations should always be shorter than the time it would take a touch-typist to repeat a missed key (about 0.1 to 0.2 seconds). Response time (time elapsed to the beginning of responses) for more lengthy instructions should be a small fraction of the time taken to specify the operation, and should never exceed about three seconds. Note that it is only necessary that the response *begin* within that time.

The Fallacy of Human Homogeneity

Finally, the battle may not be won even if a system provides excellent interactive prompting for a naïve user. The needs of a new user are rarely the same as the needs of an experienced user. Interactive prompts that are a necessity for a new user may be a frustrating waste of time for an experienced worker who is using the terminal extensively. As a further complication, the type of display device in a terminal may affect people's acceptance of instructions which are superfluous to their needs. The relatively slow output rate of a printing terminal, for example, can be particularly exasperating if most of the printing consists of instructions the user does not need. The same instructions on a video display might be perfectly acceptable since the rapid rate of display would outweigh the fact that some of the instructions were superfluous.

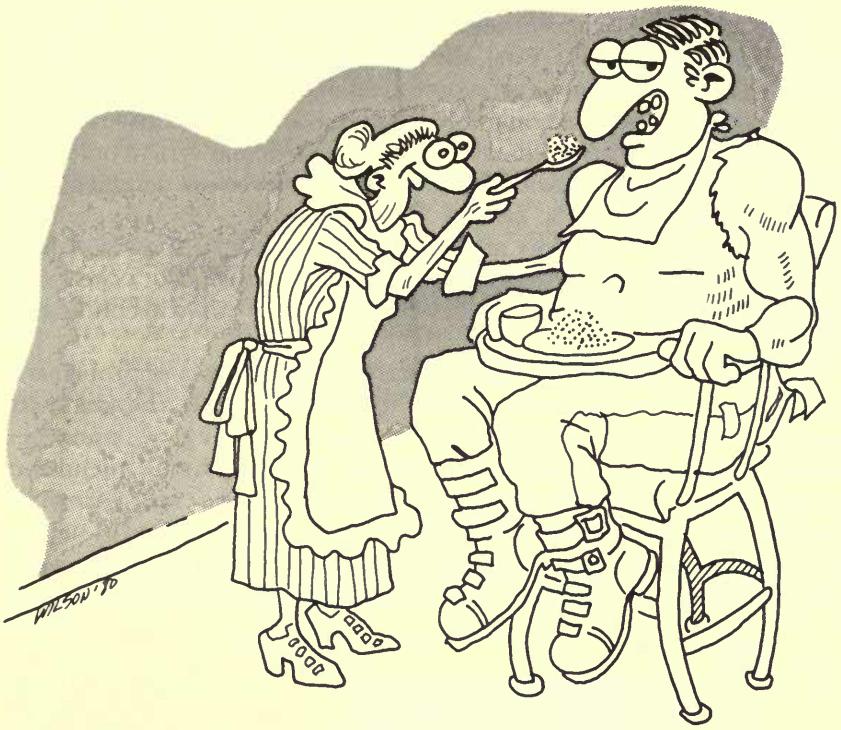

Figure 3. User capabilities usually change with time and experience.

Design of human-machine interactions must, in short, take into consideration the needs of the novice user (who will require aid at every step), the experienced occasional user (who will need minimal prompting), and the experienced user with a substantial workload (who will be mostly interested in rapid response and minimal hindrance in getting the job done). These three levels of experience are frequently telescoped in time for a given individual who sits down at a terminal as a novice and stands up (several hours later) as an accomplished user. The well-designed system must accommodate all of these levels of expertise by an appropriate mixture of optional paths, self-selected "help" sequences, and careful human engineering. The human engineering must, above all, minimize idiosyncratic forms of interactions that simplify the work of a computer programmer at the expense of the convenience of users.

Conclusion

This paper has concentrated on viewpoints rather than details of technique for two reasons. First, the physical design of systems is rapidly changing as new components become available. For example, a few years ago it would have been reasonable to list the advantages and disadvantages of making the application characteristics of a particular system hardware-resident rather than software-resident (e.g., a keyboard designed for a specific application *v.* a general keyboard with software prompts). Changes in types of memory and display devices available are now blurring such distinctions. In general, specific suggestions about system configurations simply do not "age" well in times of rapid technological change.

Second, our experience has shown that the real source of problems in most design efforts has been failure to identify clearly the goals and procedures that define the system. In the absence of clear goals, computer-design specialists must substitute their own view of what is intended or needed. To the extent that these specialists have specific experience in the practical problems of a given application, their views may lead to successful designs. To the extent that these specialists rely on the fallacies described here, the designs may be dramatically unsuccessful. As in any

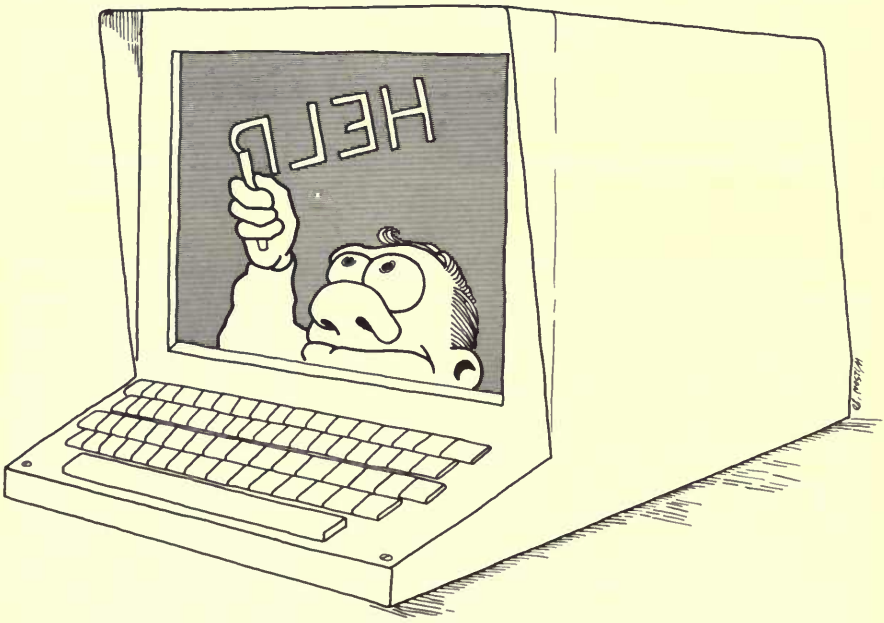

Figure 4. System designers can rarely predict all of the problems likely to be seen in a given application.

change in work procedures, there is a need for direct, active input by those who have experienced the reality of the task environment. When the change in work procedures requires investments in time and money of the magnitude demanded by selection or development of computer systems, that need becomes crucial.

The fallacies of system design described here can be averted most easily by continual, careful cooperation between design specialists and those who are thoroughly familiar with the ultimate application of the system. More than in any other form of computer system design, systems that provide interactive terminals for occasional use by minimally trained persons demand careful design to insure that expected performance occurs under realistic conditions. Systems that make unrealistic demands on user training, memory, or ability will not be truly successful even though they may function under ideal conditions.

REFERENCE

1. Broadbent, Donald A. "The Magic Number Seven After Fifteen Years." *In* Alan Kennedy and Alan Wilkes, eds. *Studies in Long Term Memory.* New York, Wiley, 1975, pp. 3-18.