ASIS Annual Meeting
San Francisco - Oct. 1971

THERE AIN'T NO USER SCIENCE

A Tongue-in-Cheek Discussion of Interactive Systems

Jacques Vallee
Institute for the Future
Menlo Park, California

The current complexities of human interaction with computers can often be traced to poor design and to misunderstandings of the user's motivations and work patterns rather than to genuine scientific problems. Therefore, many problems related to poor user acceptance of computer systems do not suggest the need for a new branch of computer science, but for an examination of human factors from a psychological and sociological point of view. If there is a genuine "user science," it will have to be based on a much more sophisticated model of human thought processes than is currently available to systems designers.

The author draws from examples of man/system interaction in other fields to support the view that better standards, serious design guidelines, careful evaluation under real-world conditions, and plain common sense would remove many of the existing obstacles for the computer user.

I. INTRODUCTION

My role in this session is that of a black sheep. My objective is to raise some hidden issues and to question some of the obvious assumptions we are all making about user science. Computers can be expected to be used increasingly by nonspecialists; yet, the interaction of this new user population with the systems we develop is not well understood; a need exists for a thorough analysis of man/machine systems. I am not questioning this need; but I am reluctant to see it glorified with the name "user science," at least until some of the major pitfalls have been recognized.

In the last three years, a small group of us has had the opportunity to observe interaction between several hundred nonspecialists and various software systems used for "conferencing." Many of the participants in these conferences had never even used a computer terminal before, and it was our responsibility to train them in everything from the setting of the switches on the device to the log-in procedure on the various networks they would be accessing, along with the various commands available to them through the program. In addition, some of them had to learn certain elementary operations of file management.

It has been an eye-opening experience for us to play this role, because the members of our group, who had varying degrees of

previous acquaintance with computer systems, had lost their sense of perspective regarding the constraints placed by systems designers on the dialogue between the machine and its users. We knew that this interaction was a much-neglected area of systems programming; but we had never realized just how cumbersome, preposterous, or plainly stupid the interfaces were.

Observing the user's frustration with such programs, outsiders might well conclude that an extremely complex area of research has been uncovered and that it represents a new science. They might even approve of research funds being spent to understand why nobody uses the information systems that research funds have been spent to implement. I will argue that such a conclusion is wrong, that it only lends dignity to a state of confusion that was unnecessarily *created by programmers* and should be *cleaned up by programmers*. If anything needs to be researched, it is not the user but the social structure that surrounds any information system, be it a set of rules and laws, a library, or a computer-based facility. It is this social structure which dictates the interface, and to focus attention solely on the problems of the user is to miss the real issue.

II. WHY DON'T PEOPLE USE COMPUTERS?

Let me offer a simple starting point, suggested to me by my colleague Hubert Lipinski: "Why don't people use computers?" *We* use computers, of course, and many of our friends do. But we represent a very small minority. We depend on computers for our work and for the information that guides many decisions we make. But people outside this community do *not* use computers in spite of the valuable services they can provide. Not only is the man-in-the-street disgusted, intimidated, awed, or repelled by computers, but many professionals view information processing with hostility. The reasons are not buried deep in the dark recesses of the human mind. They are plain and simple. They begin with the terminal, which has a plethora of design problems. They grow with the difficulty of logging into to a network or to a central computer. They blossom as soon as humans begin interacting with a piece of software.

The Terminal. Figure 1 shows a standard terminal keyboard. This is the kind of equipment our project has been shipping to schools, research institutions, and government facilities around the country to train *people* (not programmers) to use our software. Let us assume that the terminal is provided with paper, is in perfect working order, and that all the settings are correct. (I know this may sound trivial to you, but have you tried to explain to someone *over the phone* how to load a roll of heat-sensitive paper through a slit on a portable terminal? Do you know why there is an INTERNAL switch on a device that has a built-in coupler?)

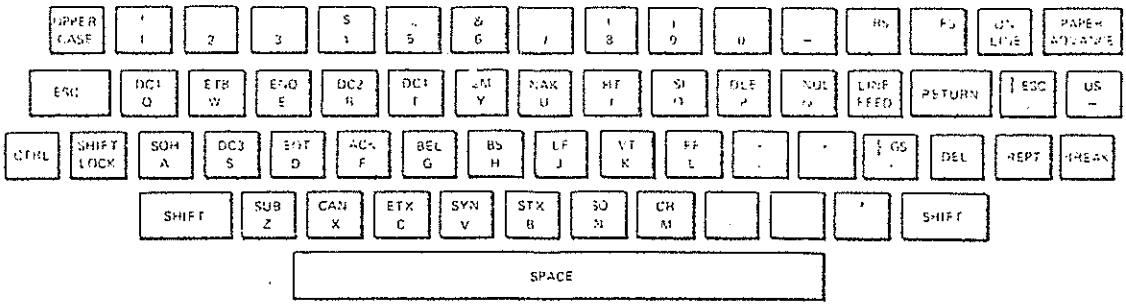

Figure 1. Poor Keyboard Design

Any *ordinary* person looking at this keyboard will assume that hitting SHIFT-E will result in the device typing the combination "ENQ," because it appears on top of the E key. Thus, many of our users go through life without ever typing a capital letter for fear of unleashing some strange combination of letters or of starting some uncontrollable chain reaction. Of course, *extra-ordinary* persons like us know better. We know that SHIFT-E on this machine will type a normal capital E and that to obtain ENQ one has to hit CTRL-E. *It is knowledge like this that makes us extraordinary.* (I confess I have no idea of what would be the circumstances under which I might need to utter ENQ, and I would be honored to meet someone who does.)

The presence of these strange-sounding codes on the keyboard is a small problem. Worse is the fact that touch typing is inhibited by the shape and position of the keys, the variable-delay echoing, and the "roll-over" feature that prevents depressing more than one key at a time. These features effectively decrease the abilities of the proficient typists.

Other problems associated with various terminals are: the glare on the keyboard; the glare of the plastic plate which covers the paper; the printing head which often obstructs the user's view of the last few characters; the placement of the keys; and the intimidating row of function keys such as INT, BRK, or READY which have names that bear a very distant relationship to what happens when you push them.

The Log-In Game. We now peep into the living room where Mr. and Mrs. Average User, having finished dinner, decide to join their favorite computer network, as they have done every day for the last two years. Although the network has thus recorded nearly 800 interactions with them, it apparently has no way to remember their real names, so they have to be known by the code "CROWN." (This is a thrill to Mrs. Average User, who is a spy movie fan, but Mr. User can never remember the password.) Figure 2 shows what happens between the time when they hear the

BEEP on their telephone and the time when the terminal types out something directly related to their interest, namely the message "WELCOME." We have repeated this operation under four separate commercial networks, which apparently stay in business in spite of their log-in procedures. (It would be unfair to this audience to include military or academic computer systems as examples. Their users have no choice.)

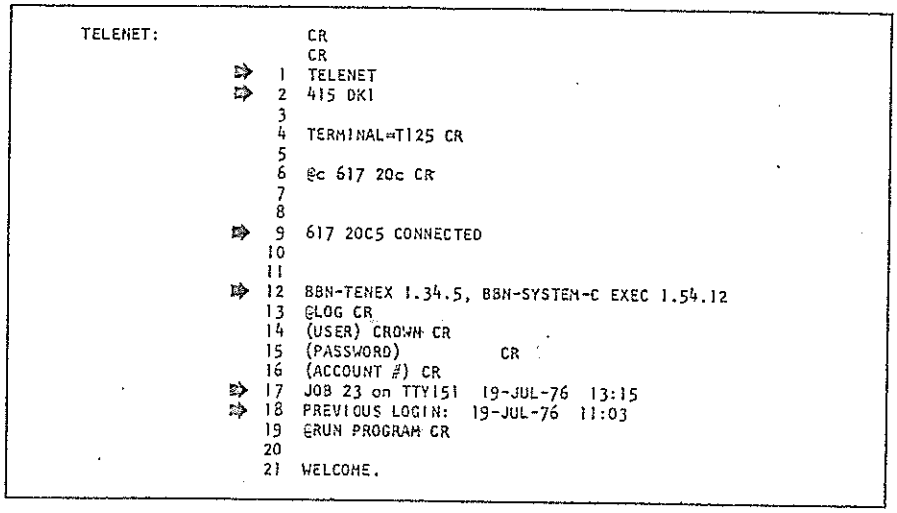

Figure 2. Log-In Procedure on A Commercial Network. The arrows indicate entire lines that are meaningless to most users. The shaded area indicates information typed in by the user.

For each interaction case, we have recorded three simple quantities:

α: the number of lines appearing on the terminal before Mr. and Mrs. Average User see something meaningful;

β: the number of keystrokes they need to type; and

δ: the number of characters typed by the system having no meaning or relevance to their interest.

The reader will see in Table 1 that even on the most expeditious network, INFONET, it takes 29 keystrokes to do nothing. (How would you like to dial 29 digits every time you want to call the girl next door? Telephone companies go out of business over things like that. Computer users go out of their minds.) On TELENET, it takes 48 strokes to do nothing; but one is rewarded with a deluge of messages in reply, ranging from "415 DK1" to "JOB 23 on TTY151" with rather obscure meanings, especially if you know what a TTY looks like. (Note that following TERMINAL=, the user replies TI25. This is a code that stands for TI725!)

TABLE 1. OBFUSCATION PARAMETERS IN FOUR COMMERCIAL NETWORKS

	TELENET	TYMNET	CYBERNET	INFONET
α:	20	10	14	8
β:	48	35	46	29
δ:	116	10	95	54

The Dialogue. Now that Mr. and Mrs. Average User have won the log-in game, they can run their favorite program. They have graduated to the wonderful world of software where anything can happen. They could, for instance, get the message "DRUM FULL," after which the terminal will refuse to do anything. Mr. Average User, who didn't even know he *had* a drum, is extremely impressed. (I recall being introduced to a medical researcher in Europe a few months after a teleconference through which we had first "met." He was a calm Britisher whose first words to me, delivered in a kind but reproachful tone, were, "You know your message that says 'HOST DIED'? Well, wouldn't it be more kind to say 'HOST PASSED AWAY'?")

Indeed it is in death that computers come closest to imitating humans. The phenomena that accompany their last few seconds are as sad, emotional, and messy as those of their human masters. In their bereavement, Mr. and Mrs. Average User might well despair of ever finding their computer healthy again, as was the case with my friend, Bob Johansen, when he lost a program amidst much typing of strange characters (Figure 3). Since he had come to the Institute from a background in the sociology of religion, he assumed that the system was speaking in tongues. Yet, when he tried to reestablish contact, he was told that "host" was no longer responding.

colorful tales. Every network has its own way of rejecting you, of pulling you into strange traps, of making you wait or repeat what you just said, of dying. Although such peculiarities can be fun for the boys and girls in the machine room, they have a devastating effect on people who are trying to get a job done. When the job involves communication and joint effort among groups located all over the Western hemisphere, as our teleconferences sometimes do, the results can be disastrous. Motivating the users again after every failure is a difficult and frustrating task. When we performed an analysis of questionnaires returned by participants in some early conferences, we found that two factors far outweighed all others in accounting for their reactions to our system. These two factors were *difficulties with terminals* and *fear of a network failure.*

III. THE OBFUSCATION IMPERATIVE

If there were a user science, the Obfuscation Imperative would constitute its first law. There is a short but interesting diagnostic on TELENET. It reads: "SUBPROCESS UNAVAILABLE." The puzzled user refers to the section called "Explanation of Network Messages" of the user manual and reads:

> The specific host process included as part of the address is not available.

Note that the so-called "explanation" has now unnecessarily added two new sources of confusion, the term "host" and the word "address." The user has no recourse but to make an appointment with a system expert. The expert knows the true meaning of the message: the computer is down! Of such knowledge expertise is made. This is not user science. This is *obfuscatology*, the science of hiding things away from other people. It reaches a peak with system completion code 213 on the IBM 360. If your program dies with "COMPLETION CODE 213-04," you must consult the appropriate manual under IEC 132 I. The system expert who owns the manual will then read the expanded text of the explanation to you:

> The format 1DSCB for the data-set could not be found on the first volume (or the volume indexed by the volume sequence number) specified by the DD statement, or an I/O error occurred reading the F-1 DSCB for the data-set.

What the message means is simply: "DATA-SET WAS NOT THERE." Why didn't they say this in the first place? The answer has nothing to do with the computer: this message has exactly the same length as "COMPLETION CODE 213-04," and the machine couldn't care less what the message says.

Is such evidence of deliberate obfuscation limited to our field? On the contrary, *it can be found whenever a community of*

specialists attempts to protect a privilege. For example, in the 13th century, a surgeon named Arnold of Villanova recommended to his colleagues to preserve their linguistic distance under the greatest diagnostic stress:

> Say that the patient has an obstruction of the liver, and particularly use the word *"obstruction"* because they do not understand what it means, and it helps greatly that a term is not understood by the people.

Examining current medical literature, Michael Chrichton concludes that contemporary physicians are still following this rule: they are trying to "astound and mystify the reader with a dazzling display of knowledge and scientific acumen." He also observes that most doctors ignore papers outside their own specialties because they can't understand them. Does medicine need a user science?

The legal profession, too, follows the obfuscation rule. When I bought my house, I signed a paper which reads:

> This Deed of Trust applies to, inures to the benefit of, and binds all parties hereto, their heirs, legatees, devisees, administrators, executors, successors, and assigns.

Even photographers obfuscate. Witness this extract from the user manual for my modest camera:

> Using the flashmatic system, after you set the guide number, the correct exposure is automatically calculated as you focus. No more fumbling through lengthy calculations to find the proper F/stop: set the mark on the F/stop ring to the center index. Obtain the proper guide number by multiplying F/stop number times distance. For example, if ASA 80 film is being used, set the film speed on the calculator to ASA 80 and check the proper F/stop number at a distance of 10 feet.

In a common practice of software design, the same operation is given different names, depending on the particular subsystem! Here again, better training or a more detailed model of the user will not help. What is needed is a thorough cleanup of the command language. Under the KRONOS operating system of Control Data Corporation, for example, there is a command called "LIST" which produces a listing of a file. Having obtained a listing, the user might want to edit the file. Under the EDITOR command, however, the LIST command becomes invalid. Only an expert will be able to tell you that the proper command to use now to produce the *same* listing of the *same* file is PRINT. But once you get out of the EDITOR command, naturally, PRINT is no longer recognized!

We could devote a whole session or a whole conference to such examples.

IV. THE WIDE ANGLE FALLACY

If there were a user science, its second law could be called the Wide Angle Fallacy. When a disgusted user goes back to the designer with the statement, "Your system doesn't perform the special function I need," the designer's ego is deeply affected. To regain the good graces of his customer--and to reestablish his or her own self-esteem--the designer is likely to answer, "I can fix it. I will add another command for you."

Later, the same designer will be seen at conventions, meetings, and workshops, extolling the virtues of his system, the "power" of which can be measured by the great number of commands it can execute. I believe this is often a fallacy and that both designers and users should recognize it. There is a similar fallacy in astronomy, related to the size of a telescope. Most novices and many astronomy students believe that you see more in a big telescope than in a little one. The opposite is true, of course. Increasing the diameter of the telescope may collect more light, but it narrows the field of view. The analogy with software design is appropriate. The early versions of our FORUM conferencing system had dozens of commands, hence a lot of potential "power"; but few users could remember the whole structure. Careful monitoring of usage pointed to ways of simplifying it, and we started looking for opportunities to cut down the list of commands. For the last two years, we have been offering a teleconferencing service which gives the user a repertory of only six commands. The potential user population (which is analogous to the "field of view" of a telescope) is now much larger. At the same time, we have found that no serious loss in "power" had been experienced. On the contrary, the new, simpler commands correspond better to the basic primitives of the interaction we are trying to support.

What is true of systems programmers applies equally well to librarians who think they are expanding the "power" of an index when they add more keywords and more terms into it. The whole issue of how to support the process of discovery instead of mimicking its side effects lies solidly buried under dozens of documentation systems which our profession is accumulating as a buffer between the scientist and his data.

V. IF NOT A USER SCIENCE, WHAT THEN?

These observations do not make me feel an urgent need for a new science that would study the user's relationship with computer systems where even the most obvious steps to human interactions have been neglected. I will argue later that a real user

science exists, but it is not to be found at this level. What we need instead is to follow a few guiding principles which will place those responsible for a system in a position to anticipate many user frustrations:

1. The first principle would be never to start implementing a system until the end users have been identified and given easy access to the designers.

2. The second principle would be to monitor everything, noting, however, that not all information is quantitative and that one cannot evaluate the potential impact of *missing* features.

3. A third principle would be to release systems to "real users" as soon as their utility is apparent to them. A real user is characterized by the consequences for the user and for the designer if the service is not performed, by the fact that the service is funded with money which could be used in other ways and which comes from an operational (versus research or exploratory) budget, and by an ultimate evaluation of the service in terms of an external outcome.

4. The fourth principle would be never to demand the user to type an input that is not relevant to the task at hand and never to give him an output that is outside the task context. (Our software now tries to intercept all system panic messages, replacing them with the statement: "Sorry, we are having trouble with the computer.")

Attempts to make computer professionals aware of the need for these elementary guidelines often end up in frustration. They assume that if you raise such a question, you can only do so out of either resentment or ignorance. At best, they will send you on your way with an elementary tutorial on operating systems and a PL/1 manual. This attitude was illustrated to us by a recent discussion between Thad Wilson, a member of the Institute staff who has used computers for 15 years, and the manager of the Northern California facility for a leading network company:

Wilson: "Many of our users do not understand why the system sometimes suddenly stops and says, 'PLEASE LOG-IN.' How are they supposed to understand that? *They've already logged in.*"

Manager (trying to confuse Wilson): "They need to log-in again because a failure has occurred. It could be the node, the supervisor, the host, or the network."

Wilson (not confused): "Why don't you simply tell them that you're sorry that service will be interrupted for a while?"

Manager: "You don't realize how long it takes us to change something. We've been in business a long time. You should train your users better and give them our manuals."

Wilson: "I don't think it's a good idea. You should change the system instead."

That actual discussion illustrates a sad point: when we encounter problems with the use of computer systems, we can change either the system or the people. What is frightening is that the computer industry probably has enough power now to start changing people. This network manager has already been changed, and he is working hard to change others. The programmers under him work hard to become like him. To modify the system is too difficult for them to even consider. The reasons they give are not technical, but social and bureaucratic. After all, they "have been in business a long time." Somebody would have to rewrite all those forms and all those manuals.

The social environment and the anecdotes surrounding the marketing, maintenance, and breakdowns of computer products should also come under the attention of would-be user scientists. The vast distance between the services delivered by the computer community and the users it pretends to serve was made obvious to us in the course of a computer conference which included as a participant Mr. Richard Bach, the author of *Jonathan Livingston Seagull*, who lives in Florida. After a period of unusual silence from him, we learned that his terminal had broken down. We tried unsuccessfully to put pressure on network and terminal suppliers to try to help him. When contact was reestablished (a week later), he gave us some of the details:

Private Message from BACH 1-Jul-75 7:08 PM
Some notes in the quiet here about events during the breakdown last week of my terminal. The interesting learning for me is that an individual with his terminal inoperative is the low priority item in the system. Good reasons, and all that, but the feeling in the computer terminal world is that good excuses are acceptable, which would be rare in a high-competition field.

It would be most interesting to analyze user profiles, command usage, and interaction patterns when computers begin to be used by writers and communicators of the caliber of Mr. Bach. But first we must find a way to provide them with paper for their terminals and give them reliable access to systems they can use. This simple prerequisite has not yet been met by the computer community.

The examples quoted above have attempted to show that situations frustrating to the nonspecialist user have often been deliberately created to protect a self-styled elite of programmers. A massive effort to redefine the job control language (JCL) of IBM, for instance, would free the creativity of thousands of people and expand the market for many computer services; but at the same time, it would make obsolete a mass of folklore and machine room recipes that passes for knowledge and creates job security for programmers throughout the world. The Obfuscation Imperative is a clever marketing strategy. Someone trying to understand the relationship of the user to the system has to first realize that the situation is dictated by social, psychological, economic, and professional constraints. Only when elementary design principles--such as those we have only begun to outline here--have been applied to software interfaces will we be able to really speak of a genuine science.

The topics which could be studied as part of a design science are varied and exciting. They are already found in the investigation of cognitive styles (as in the work of Peter Keen at Stanford), the perception of the computer as a confidant or an adversary (as in Chris Evans' experiments in England), several artificial intelligence prototypes, mapping, interactive graphics, computer-aided instruction, and advanced data analysis packages. This experience, however, has never been pulled together into a body of knowledge; it is certainly desirable and urgent to do so, but it still leaves unanswered the question of a general user model.

In the meantime, I can only concur with William Blake who wrote in *Jerusalem* (f10.20):

I must create a system
Or be enslaved by another man's!