Creating software that people can and will use

Kathleen Potosnak,
Human Factors Editor

Human factors engineering is an applied discipline that aims to design equipment, environments, tasks, and tools to accommodate human needs and limitations. Human factors views a design from the user's perspective. The emphasis is on how the user will operate the system, not on how to get the system to work internally.

Software human factors is concerned with what makes computers easy or hard to use, how people use computers, and how to incorporate human factors into the design process. Human factors engineers have developed methods, tools and guidelines to improve software usability. They conduct experiments with real users and apply that knowledge to the design of real systems.

A rose is a rose . . . Another name for human factors is ergonomics, which comes from the Greek "ergos" meaning "work" and "nomos" meaning "laws" — the laws of work. The word was imported from Europe primarily by the furniture industry. For this reason it has come to be associated in the US with aspects of the physical work environment.

The American name for this discipline has changed over the years. "Human factors engineering" has become the most common name. Less frequently used are "human engineering" and "engineering psychology."

A bit of history. Human factors engineering got its start during World War II. In the early 1940s, US Air Force pilots were having many accidents attributed to "pilot error." The government hired psychologists to study the pilots and to find out what was wrong with them. After testing and observing the pilots during maneuvers, the psychologists concluded that nothing was wrong with them. The problem was with the design of the planes!

As one psychologist put it, "The realization gradually began to form that the best efforts of selection and training specialists were often negated by the way equipment was designed. The realization came slowly because for years everyone had been conditioned to attribute most accidents to 'human error.' It took a long time for us to discover that 'human error' could be caused by or diminished by the way equipment was designed" (A. Chapanis, "A Psychology for Our Technological Society, or, A Tale of Two Laboratories," in One Hundred Years of Psychological Research in America, S.H. Hulse and B.F. Green, Jr., eds., Johns Hopkins University Press, Baltimore, 1986, p. 57).

At that time, each pilot had to fly all sorts of different planes. There were so many inconsistencies in the designs of the controls and displays that, according to the psychologists, it's a wonder there weren't more accidents than there actually were.

Just as pilot performance is affected by inconsistent aircraft designs, user performance is affected by incompatibilities in the ways software programs operate. Most of us have experienced the frustration of switching between two (or more) text editors. The problem is compounded when you consider the full range of programs we use.

User interface. Human factors engineers consider software in the broad context of the user interface. The interface consists of everything that a user interacts with to operate a computer, including software, hardware, task procedures, documentation, and training materials.

Because these aspects are all related to each other, it is difficult to evaluate the ergonomics of a software program in isolation. For example, a command-language dialogue style may be just great for frequent users, but it must be supported by appropriate training and documentation to be usable. As another example, consider a graphics package that requires the user to press Ctrl with another key for every cursor movement (as in Wordstar): It would make drawing very difficult.

Functionality is not enough. What makes software ergonomic? Most software vendors emphasize their products' functions. They produce numerous charts comparing the functions of their competitors' products. Announcements of upgrades always list the additional functions implemented in the new versions.

Although functionality is important, it doesn't always hold true that the more functions, the better the software. Usefulness and usability are the keys to designing software that people can and will use.

Usefulness. A program can provide hundreds of functions, but if they aren't useful, the software won't be successful. Useful software provides the functions people need to get the task done.

To determine which functions, you should study how people actually perform their tasks — whether with or without computers. The closer the match between how users think about the task and the functions provided, the
more ergonomic the software is.

The success of spreadsheet programs is a good example of the benefits of matching the software to how people think about their tasks. Spreadsheet programs are useful because they let people do the same task in pretty much the same way — but faster than — they could do it on paper.

Usability. Even if the system provides useful functions, there is still no guarantee that people will use them. Imagine two spreadsheets — one where you type the numbers directly into the cells of the spreadsheet and another one that requires you to access the data from a separate file by typing code names on a command line at the bottom of the screen. Which one would you use more often?

Usability is paramount to designing software that will be used. Sure, we've all put up with crummy software when we had to get a job done and there was no other way to do it. But this situation is costly, time-consuming, and increasingly unattractive to users.

Reactions to poor design. People have fairly typical reactions to systems lacking in usefulness or usability:

- Confusion. Difficult programs confuse people. Users have to figure out what the system is trying to tell them, so they take longer to complete their tasks. Confusion can also lead to other, more serious reactions (such as those that follow).
- Frustration. People give up easily or become angry with the system when it seems that they are being asked to do something that they are not in fact supposed to do. How many times have you threatened to throw yours out the window?
- Panic. Slow or unreliable systems cause users to panic when they are under pressure to get the job done. People also panic when they perform (or think they performed) a catastrophic action. I'll tell you I was panic-stricken the day I mistakenly erased three months of work because I didn't understand a cryptic warning message. (I almost went out the window that time.)
- Boredom. The user's mind wanders when the system is too slow or the task is too simple. Boredom leads to increased error rates. Many companies have to create special motivation and incentive programs to keep their data-entry operators on the job.
- System abandonment. People with the discretion to reject the system will find other ways to perform their tasks. For example, an evaluation of a financial-management system revealed that although the system had been in use for several years, some of the departments used their own computers to perform the same functions the central system provided. Other departments went back to a manual (noncomputerized) system! (Malde reported this in "How a Real-Life System Stands up to the Commandments," *Behaviour and Information Technology,* Jan.-March 1986, pp. 81-87.)
- Incomplete system use. The most common reaction to a difficult system is to ignore part of its capabilities. For instance, the staff of a major clearing bank was given an on-line inquiry system with 36 functions (accessed as codes). But only four codes accounted for 75 percent of system use. Consequently, there were costs both in computing time and user time, as well as an increased possibility of user error. (Eason reported this in "Towards the Experimental Study of Usability," *Behaviour and Information Technology,* April-June 1984, pp. 133-143.)
- Indirect system use. When they find it too difficult to use the system themselves, many users will try to find someone else to do it for them. For instance, although I have access to an airline reservation system, I pick up the phone and ask my travel agent to make my arrangements.
- Task modification. When the system doesn't match what people really need to do, they may change the task so it matches what the system can do for them. A manager's comments about a computerized stock-control system illustrate the point: "They didn't ask us what we wanted and then write the software to order. They produced the software, and we had to change to accommodate it. The ordering system was working fine and the accounts people were coping. The one thing we needed was a decent requisition system. But we didn't need a computer to get that." (Rowe reported this in "Identifying Causes of Failure: a Case Study in Computerized Stock Control," *Behaviour and Information Technology,* Jan.-March 1985, pp. 63-72.)

- Compensatory activity. To take advantage of the system's capabilities, people may have to do extra work by hand, such as reformatting information to match the structure required by the computer. This is exactly what the users at the clearing bank wound up doing.
- System misuse. Users with enough knowledge of the system may circumvent the rules and change the system to meet their personal requirements. For example, assigning higher priority to your own work on a time-sharing system is a common reaction to slow response times.
- System modification. Sophisticated users may decide to program the system themselves to get it to do what they need done. Computer hackers are notorious for doing this. When users modify the system to suit themselves, data security and maintenance efforts may be compromised.

Complaints are real. The complaints of computer users are real. And so are their reactions to poor system design. To make software tools that are accessible to people, system designers must find a match between

- the users,
- their tasks,
- the equipment, and
- the environment.

System designers need a great deal of information about how their systems will be used and by whom. Human factors provides this information about users — their abilities, limitations, behaviors, and preferences — and their tasks. Quantitative measures of performance speed and accuracy support the design decisions. Recommendations are based on research and knowledge of how people actually use things, not on conjecture or personal opinion.

Human factors engineers also use formal methods to apply this information to system design. This is how human factors bridges the gap between what machines can do and what people want to do with them.

Not to be confused with common sense. Because the findings of human factors research often make good sense, many people initially believe that ergonomics is little more than common sense. However, human factors engineering is not just common sense. It is a formal, applied, and multidisciplinary profession with strong ties to psychology and engineering.

If human factors were just a matter of common sense, I'd have to conclude, as I look at some of the products that we use every day, that common sense just isn't all that common.