Edgar H. Sibley
Panel Editor

Instead of limiting functionality, usability complements functionality. It affects how and with what effectiveness a system is used, and even whether or not it is used at all.

FUNCTIONALITY AND USABILITY

NANCY C. GOODWIN

Too often designers of computer systems equate functionality with usability or view usability features as limiting functionality. Certainly, it is important that a system provide the functions a user needs to accomplish a task or set of tasks. However, it is a mistake to suppose that design features intended to enhance usability are niceties to be provided at the designer's convenience, and that if a trade-off is to be made it should be made in favor of functionality. There is increasing evidence that the effective functionality of a system depends on its usability.

After some comments about system functionality and usability, and why designers may avoid focusing on usability, this article will cite and discuss evidence from the literature demonstrating that functionality and usability are complementary system characteristics.

WHAT IS FUNCTIONALITY?

The need for functionality is obvious. Users will select systems that provide functions needed to do their tasks. They will not select a reservations system to manage banking tasks, or a spreadsheet to do word processing.

The task of specifying requirements for system functionality seems straightforward. A designer negotiates with users or their representatives a list of desired functions and then provides those functions. By comparing the list of functions requested by the users to the list of functions provided by the system, the designer knows how well the system will meet users' needs.

However, people's reasons for choosing to use a computer system in the first place may differ. One rationale for using a computer system is because it is the only way to get a particular job done. There are many high-volume, highly structured tasks that depend on system use—reservations systems, insurance form processing, catalog sales, banking, and telephone directory services—where, once a system has been introduced, it is hard to imagine returning to the old manual methods.

Another motivation for computer use is to help someone do a job better or faster. In these cases, the tasks are generally less structured, and computer use is more discretionary; whether or not a user considers a computer *necessary* for these jobs depends on how well the computer meets the user's needs. Many office applications, decision support systems, and information retrieval systems fall in this category.

There often exists a perception that the more functions are provided, and the more flexibility and the more complexity in the system, the better. However, for both discretionary and nondiscretionary users, *the way* in which the functions are implemented will have a significant impact on system usability.

In [22], Nickerson categorizes the complaints that cause people to avoid using a particular computer system: limited functionality, accessibility–availability problems, start–stop hassles, poor system dynamics and response time, work-session interrupts, inadequacies in training and user aids, documentation, command languages, consistency and integration, and users' conceptualization of a system. Note that functionality is only one of the factors influencing user acceptance, most of which relate to how the system can be used rather than whether or not a particular function is available.

WHAT IS USABILITY?

Usability is less easily defined. It is affected by the types of tasks to be accomplished: A keyboard-based interface appropriate for a word-processing application may be inadequate for a graphics application. In this respect, usability, like functionality, is task related; it is also people related. The characteristics that make a system usable for one set of users may render it unusable for another. First-time, casual, and expert users may all have different requirements, and their requirements may change as they move from one level of expertise to another.

For those interested in a definitive or complete discussion of usability, there is a wealth of research, reports, and articles whose aim is to describe what makes a system usable. In a fine survey of current work, Paxton and Turner [23] focus on the novice user, although many of the points they raise would also apply to experts.

Although we are making progress in specifying usability in measurable terms, it is still too often discussed in abstract terms. A designer trying to find out how to design a usable system may find in a technical journal a statement like this:

> To be truly usable a system must be compatible not only with the characteristics of human perception and action, but, and most critically, with users' cognitive skills in communication, understanding, memory, and problem solving. [1]

Or, turning to a magazine, the designer might read, "A friendly system has three important aspects. It is cooperative, preventive, and conducive" [18]. Although such abstract conceptualizations may be quite valid, they do not offer specific guidance.

HOW CAN USABILITY BE ACHIEVED?

Some of the factors affecting usability are organizational and may be beyond the designer's control. Training, accessibility of terminals, and the culture of the workplace all have an impact. The entire population of users must be accommodated, whether they are first-time, casual, or expert users, or represent a combination of different levels of expertise. Some computer system characteristics, such as slow response times during periods of heavy usage, may also be beyond the designer's control.

Other factors affecting usability are more directly within the designer's purview. The designer can provide functions that match task requirements, and determine the details of screen design, command language or menu interaction techniques, system feedback, and the dynamics of user–system interaction.

There are increasing efforts under way to develop techniques for including usability goals in the design process. Carroll and Rosson [5] focus on developing measurable usability specifications to be used with an iterative design process, while Eason [8] discusses variables affecting usability in the overall context of system use: that is, looking beyond laboratory experiments aimed at evaluating specific features to an assessment of overall system impact on job performance.

Advice on designing usable systems proliferates: from brief articles in popular magazines [18, 21, 24], to books [11, 16], journal articles [6], [19], and reports [26]. However, even guidance written explicitly for designers may need informed interpretation. For example, of the 679 guidelines discussed in the Smith and Mosier report [26], most are generally stated and must be transformed into specific rules for application to a particular system design [15, 20].

HOW USABILITY AFFECTS FUNCTIONALITY

Opinions on the importance of usability in system design are not particularly new or unanimous. Martin [16] has written that a user's ability to use a system powerfully will depend on the ease with which he or she can communicate with it. Brooks [3] considers usability "the proper criterion for success," and Bennett [2] argues that "user acceptance is strongly affected by *how* the function is invoked as well as *what* function the system contains." Foley and Van Dam [9] conclude that usability is at least as important as functionality.

On the other hand, Fried [10] cautions us that "in general, there is little hard evidence to support the idea that ease of use leads to improved (traditional) productivity, or that specific ease-of-use characteristics truly make software easier to use for a majority of users." In a sense, functionality itself can determine usability; if the functions provided do not match task requirements, a system will not be usable.

There *is* a growing body of evidence that shows that providing extensive functionality is not enough: People must understand what the functions do and how to use them. In [3], Brooks describes the unpublished work of W. B. Wright, who developed a system to display and study proteins—a system that required almost 100 commands to satisfy one of its applications. However, by analyzing the commands, it was possible to cluster them into small, meaningful subsets that could be used to complete each conceptual task. Those subsets were presented in menus structured to minimize menu changes during task completion.

Although Brooks cites this system as an example of the conflict between power (many commands) and ease of use (few commands), it is an excellent example of how good interface design can enhance functionality. None of the functions were removed from the system, but they were made easier for users to find and select. By grouping together on a menu the commands needed for a task, the user is not required to learn (and select from memory) a large functionally disparate group of commands. Instead, the user is able to select commands from a small related set.

Unfortunately, a designer's task is not always as straightforward as grouping related commands into menus. Meadows [17] studied users of a database search system from their first encounter through mastery of the system, looking at three different levels of prior computer experience and three different levels of language complexity. Each language provided the functions needed for users to complete the search tasks: The simplest language was menu based; the more complex language enabled users to enter commands; and the most complex language provided command entry and Boolean search capabilities.

Meadows found that the least experienced group using the least complex language, and the most experienced group using the most complex language, performed the best. Moreover, users' satisfaction with a language changed according to their experience. As users of the simplest language became experienced, they became less satisfied with that language; as users of the more complex language gained experience, they became more satisfied. These results suggest that it was not the underlying functionality that affected user performance, but the way in which users accessed those functions.

An earlier study by Walther and O'Neil [27] is sometimes cited as showing that flexibility—availability of abbreviations, default values, command synonyms, etc.—improves an expert user's performance, but hinders the novice. As published, the study does not present the actual data and does not seem to support so strong a conclusion. The confusion may be due to different definitions of the term *novice user*. Whereas a *novice* is often considered to be a person with little computer experience (i.e., a person who is not an expert), Walther and O'Neil use the term to refer to a first-time user who has never touched a computer before. The problem is that, after the first session, such users are no longer novices. According to Walther and O'Neil,

> interface flexibility is not uniformly effective with all users in optimizing performance. In a single encounter with the on-line system, users are more prone to make syntax errors if offered short-cut flexibility options. Nevertheless, most all users of the flexible version worked significantly faster than those not having the options. The exceptions were novices who worked more rapidly without the options than with them. . . . In general, users having access to flexibility options made many times more syntax errors.

Thus, regardless of user experience, flexibility leads to errors. But, once users gain just a very little experience, flexibility in a language starts to become helpful.

These studies show that there is no simple answer to user interface design. Not only do different users have different requirements, but the requirements change over time. Novice users do better with a simple language, whereas experts benefit from complexity, but the benefits of complexity have to be balanced against the cost of making errors.

Unfortunately, some designers assume that usability is only an issue for systems designed for novices. According to Shneiderman [25], even expert users are penalized by poor design: "Even expert users of interactive editing or command languages were found spending one-third of all commands in making or correcting errors." Although it may not always be possible to translate the costs of these errors on a per-error basis, it is obvious that the time required for error correction imposes costs in both staff and computer time.

For high-volume, structured tasks, improved usability can have significant, measurable effects. For computer systems used for large-scale transactions, seemingly small improvements in usability can translate into large cost savings: Saving as little as 1 second per transaction can mean a savings of thousands of dollars as well as significantly improved productivity.

In [13], Keister and Gallaway describe the scope of the productivity improvements possible through improved design. They describe how a user interface to a data-entry application was redesigned to accommodate changes in screen format and content, elimination of unnecessary abbreviations, increased consistency in wording and operational procedures, changes in error correction and feedback, and added use of on-line help. As a result, task completion times and error rates were both reduced by 25 percent, a combined savings that could reduce the workload by 17 weeks per employee per year.

While monitoring the use of a statistical package by 11 graduate students with varying degrees of experience with the package, Davis [7] found that misleading command names for the more than 100 commands (of which only 34 were used) and poor system feedback contributed to a lack of understanding of the system. Although some of the errors committed were trivial, Davis reports that a poorly chosen command name could cause serious errors: "The command SAVE FILE was widely misinterpreted, causing often disastrous and mysterious space allocation problems for the novice." Other errors caused system crashes. Moreover, because of poorly worded feedback messages, the students often reran jobs that had already been successfully completed. This is a case where poor interface design in a system providing a great deal of functionality contributed to the misuse of that functionality.

Poor interface design can have more serious consequences than misuse and errors. In [4], Conrath describes a military logistics system that ceased to be used six months after implementation because it required extensive training, had a high error rate for data input, and produced unusable output. According to Conrath, "While the software did everything asked of it, the formats of both input and output were virtually incomprehensible to anyone without a computer programming background." Consequently, users stopped using it and reverted to the telephone to get information: Any potential benefits that might have derived from the system were lost, and its functionality became irrelevant because it was unusable.

In studying the use of a banking system in which users query a database by entering a customer's account number and a code for a type of data report, Eason found that users would pick familiar codes to accomplish certain tasks even though these codes were unsuitable and more appropriate and efficient codes existed [8]. Instead of exploring system capabilities, users learned the minimum amount needed to accomplish their primary tasks: At best, they obtained more data than they needed; at worst, they obtained the wrong data or failed to find the information they needed. When the user interface was redesigned to present the codes in a more logical and accessible format, the number of codes used increased. The functionality improved not because more functions were added, but simply because they were presented in a more accessible fashion.

In a comparison of two message-handling systems, Goodwin [12] finds a similar interaction between usability and functionality. One set of message-handling capabilities was provided in a functionally rich but difficult to use system, and another in a limited but easier to use system. Although system programmers claimed that the richer system was better, preliminary data on system use indicated that, while many message-handling functions were used in the simpler system, fewer were used in the more complex system. In effect, the richer but harder system provided *less effective functionality*. If only the use of the harder system had been observed, one might have concluded that many of the secondary functions were not needed. However, the use these functions received in the simpler system showed that they were in fact valuable.

Poor usability may have more subtle effects. Long, Hammond, Barnard, and Morton [14] point out that poor usability may jeopardize the utility of a system if it causes some users to give up on it. When a computer system is used in parallel with manual procedures (i.e., during an introductory period) and the system is difficult to use, the currency of data in the system may be at risk. By opting to continue using the manual procedures rather than learning the new system, recalcitrant potential users may cause the data in the system to become out-of-date: And functionality is worthless when processing worthless data.

USABILITY DOES MATTER

Designing a usable system requires understanding the intended users, their levels of expertise, the amount of time they expect to use the system, and how their needs will change as they gain experience.

Although usability is not an easy concept, investing in usability is as important as investing in functionality. Failure to consider usability can lead to system failure. At best, a system with poor usability will cost its users time and effort; at worst, it will not be used at all, and its functions may be removed because their utility has not been demonstrated.

Usability is not just an advertiser's buzzword describing features that a designer might add or not, at his or her convenience. As an integral part of system design, usability contributes to overall system functionality by making it accessible to users and facilitating effective use of functional capabilities.

REFERENCES

1. Barnard, P.J., Hammond, N.V., Morton, J., and Long, J.B. Consistency and compatibility in human-computer dialogue. *Int. J. Man-Mach. Stud. 15*, 1 (July 1981), 87–134.
2. Bennett, J.L. Incorporating usability into system design: The opportunity for interactive computer graphics. In *Proceedings of the International Conference on Cybernetics and Society* (Tokyo-Kyoto, Japan, Nov. 3–7). Institute of Electrical and Electronics Engineers, New York, 1978, pp. 1119–1124.
3. Brooks, F.P., Jr. The computer scientist as "toolsmith"—Studies in interactive graphics. In *Information Processing 1977*, B. Gilcrist, Ed. Elsevier North-Holland, New York, 1977, pp. 625–634.
4. Conrath, D.W. Considerations for the design of office communication-information systems. In *Proceedings, Office Automation Conference* (San Francisco, Calif., Apr. 5–7). American Federation of Information Processing Societies, San Francisco, Calif., 1982, pp. 825–835.
5. Carroll, J.M., and Rosson, M.B. Usability specifications as a tool in iterative development. RC 10437 (#44642) 4/3/84, IBM, Yorktown Heights, N.Y., 1984.
6. Coulouris, G.F. Designing interactive systems for the office of the future. *Behav. Inf. Technol. 1*, 1 (Jan.–Mar. 1982), 37–42.
7. Davis, R. User error or computer error? Observations on a statistics package. *Int. J. Man-Mach. Stud. 19*, 4 (Oct. 1983), 359–376.
8. Eason, K.D. Towards the experimental study of usability. *Int. J. Man-Mach. Stud. 3*, 2 (Apr.–June 1984), 133–143.
9. Foley, J.D., and Van Dam, A. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading, Mass., 1982.
10. Fried, L. Nine principles for ergonomic software. *Datamation 28*, 11 (Nov. 1982), 163–166.
11. Galitz, W.O. *Handbook of Screen Format Design.* Q.E.D. Information Sciences, Wellesley, Mass., 1981.
12. Goodwin, N.C. Effect of interface design on usability of message handling systems. In *Proceedings of the Human Factors Society 26th Annual Meeting* (Seattle, Wash., Oct. 25–29). Human Factors Society, Santa Monica, Calif., 1982, pp. 69–73.
13. Keister, R.S., and Gallaway, G.R. Making software user friendly: An assessment of data entry performance. In *Proceedings of the Human Factors Society* (Norfolk, Va., Oct. 10–14). Human Factors Society, Santa Monica, Calif., 1983, pp. 1031–1034.
14. Long, J., Hammond, N., Barnard, P., and Morton, J. Introducing the interactive computer at work: The users' views. *Behav. Inf. Technol. 2*, 1 (Jan.–Mar. 1983), 39–106.
15. Maguire, M. An evaluation of published recommendations on the design of man-computer dialogues. *Int. J. Man-Mach. Stud. 16*, 3 (Mar. 1982), 237–261.
16. Martin, J. *Design of Man-Computer Dialogues.* Prentice-Hall, Englewood Cliffs, N.J., 1973.
17. Meadow, C.T. User adaptation in interactive information retrieval. *J. Am. Soc. Inf. Sci. 34*, 4 (July 1983), 289–291.
18. Meads, J.A. Friendly or frivolous? *Datamation 31*, 4 (1985), 96–100.
19. Morland, D.V. Human factors guidelines for terminal interface design. *Commun. ACM 26*, 7 (July 1983), 484–494.
20. Mosier, J.N., and Smith, S.L. Application of guidelines for designing user interface software. *Behav. Inf. Technol. 5*, 1 (Jan.–Mar. 1986), 39–46.
21. Moynihan, J.A. What users want. *Datamation 28*, 4 (Apr. 1982), 116–118.
22. Nickerson, R. Why interactive computer systems are sometimes not used by the people who might benefit from them. *Int. J. Man-Mach. Stud. 15*, 4 (Nov. 1981), 469–483.
23. Paxton, A.L., and Turner, E.J. The application of human factors to the need of the novice user. *Int. J. Man-Mach. Stud. 20*, 2 (Feb. 1984), 137–156.
24. Shneiderman, B. How to design with the user in mind. *Datamation 28*, 4 (Apr. 1982), 125–126.
25. Shneiderman, B. The future of interactive systems and the emergence of direct manipulation. *Behav. Inf. Technol. 1*, 3 (July–Sept. 1982), 237–256.
26. Smith, S.L., and Mosier, J.N. Design guidelines for user-system interface software. ESD-TR-84-190, MITRE Corp., Bedford, Mass., 1984.
27. Walther, G.H., and O'Neil, H.F, Jr. On-line user-computer interface—The effects of interface flexibility, terminal type, and experience on performance. In *AFIPS Conference Proceedings*, vol. 43. (Chicago, Ill., May 6–10), AFIPS Press, Reston, Va., 1974, pp. 379–384.

CR Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems—*human factors*
General Terms: Design, Human Factors, Performance
Additional Key Words and Phrases: Functionality, Usability

Received 11/85; revised 8/86; accepted 11/86

Author's Present Address: Nancy C. Goodwin, The MITRE Corporation, Burlington Road, Bedford, MA 01730.