Scientific Programming 16 (2008) 271–274
DOI 10.3233/SPR-2008-0248
IOS Press

Book Review

Stephen R. Ellis

The Design of Future Things by Donald A. Norman, Perseus Publishing, 2007, ISBN: 9780465002276

Over 25 years ago, Alan Kay, the prescient pioneer of object-oriented programming and personal information appliances, remarked that the "best way to predict the future is to invent it". But an alternative approach is to read, travel and talk with others who are inventing it. Don Norman's book, *The Design of Future Things*, is partially the product of such a venture. In it he extends his earlier work, *The Design of Everyday Things* [5], into the future. But his present book is not so much about the future, as it is about how human users can, will or should interact with the increasingly automated devices and systems they will encounter there.

From his visits to those who are inventing the future, Norman presents a broad view of future gadgets and technology, but he clearly sees that much of the activity is technology-push and will not necessarily provide a better future: for example, users of the coming technology may need to learn to deal with the frustrations of cars that discuss the traffic with each other, "talk" to the roads they travel over, and converse with their drivers who they may overrule if nearby hazards are not appropriately avoided. But Norman also realizes that the new smart products are in fact coming. They will appear not so much as Robbie-the-Robot walking androids, but will be most likely embedded in devices we already use. We will have to adapt. The intent of his book is to prepare us for this coming world. He also hopes to shape future designers' imagination so that their new products are useful, pleasant to operate, and keep their operators sufficiently engaged so that when their automation fails, the users are not totally lost.

To this end he has abstracted eleven user-interface rules to enable the coming automation to realize its intended improvements. The first five are roughly distinct, though some are interrelated and are mainly intended for interaction with automatic systems (Table 1). All are certainly easier stated than instantiated, but they do establish goals which can be investigated in specific circumstances. Moreover, they provide a concise summary of the main points of his newest book.

The first rule to provide users with rich, complex and natural signals addresses the problem that interfaces to newly automated systems often remove important, incidental sensory information provided by predecessor technology. This background information, such as the sound of water boiling telling us to dump the spaghetti into the pot or the vibration of a moving car warning us of excessive speed, provides an informational context that helps users understand the environment enveloping the actions they may take. Consequently, automated systems need to provide replacement cues. A typical example is an audio click triggered when a software-created switch is toggled on a Graphical User Interface. This sound replaces the older switch feel that confirmed switch operation.

Not only do these new systems need to provide rich and varied feedback to help signal context to the users, they also need to be sensitive to a rich and wide variety of signals that users consciously and unconsciously transmit back so as to be able to infer the users' intent. Stress detected in voice commands, for example, could signal the need for help menus to be offered. Saccadic eye movements to a switch can predict user operation of it within the second or so.

The next three rules, two through four, are interrelated. They capture the need for a user of an automated system to have an overall conceptual model. A typical model would be the desktop metaphor for most common graphical user interfaces. Systems with understandable metaphors are predictable and understandable without instruction. Users easily guess, for example, that moving a file icon into a storage device icon causes a file transfer, whereas moving it into a "garbage can" causes a file deletion. Matching the comprehensibility and utility of the desktop metaphor with equally successful new metaphors for radically different hardware is not necessarily easy, but we cannot fault Norman's rules for this. He does not promise that following the rules will be easy.

His fifth admonition to "provide continual awareness without annoyance" requires the designer to make good guesses as to what a user is expecting after interacting with a system. The user's search should not be

Table 1

Design rules for user-interfaces to automatic systems
1. Provide rich, complex, and natural signals.
2. Be predictable.
3. Provide good conceptual models.
4. Make the output understandable.
5. Provide continual awareness without annoyance.

Table 2

Design rules for machines for their interaction with users
1. Keep things simple.
2. Give people a conceptual model.
3. Give reasons.
4. Make people think they are in control.
5. Continually reassure.
6. Never label human behavior as "error". (Rule added by D. Norman, the machine's human interviewer.)

in vain. The worst thing for a user interface is for it to just sit there seemingly unaffected by user input. The input needs to be acknowledged, and, if the processing will require significant time, the expected completion time and state of activity needs to be signaled. The progress bar used to give feedback during file transfer is a classic example of this form of feedback. Bruce Tognazzini [7] notes in his design guidelines that such continual feedback is essential if the initiated process will consume more than two seconds.

Through a conceit that as part of his research for the book he has also conducted an interview with an intelligent machine, Norman additionally reports a list of rules "suggested" by machines for their interaction with their sometimes compromised human users (Table 2). These, of course, are further recommendations for the machines' designers.

It is nice to see that the intelligent machine Norman "interviewed" recognized in the first rule the need to keep things simple. Unfortunately, simplicity is neither simple nor unaffected by context or cultural background so following such a general recommendation may in fact be deeply complicated. Conceptual models discussed earlier and noted in rule two are also good for simplicity. After all, Norman already thought of this himself without having to consult a machine! The design problem, however, is the communication of the conceptual model to the user and its implementation in a consistent way. The design metaphor is usually intended to tap the users' common experience, but this intention is, of course, also intrinsically cultural and only works well if the designers share common relevant experience with their intended users. Perhaps such a mismatch is one reason why TV remote controls designed for use in the West by Asian electrical engineers are so mysterious.

The recommendation to give reasons for actions, taken or expected, is a much more securely universal rule than the others. People love to have reasons to do things and hate actions that are seemingly arbitrary or capricious. Explanations, however, need not always be presented. Indeed, if overly frequent, they can easily become obnoxious. But users' knowledge that an explanation for an action could be accessed if needed, goes a long way towards building user confidence. This principle has been known by developers of automated information systems since the early days of A.I., around the late 1970's, when intelligent medical advice systems such as Mycin provided explanations regarding suggestions for antibiotic therapy [2].

Giving users the sense they are continually in control and the provision of continual reassurance, the fourth and fifth recommendations, share some common features since a sense of control is certainly one element of reassurance. Reassurance can, of course, take many forms: from the vacuous, wiggling computer icon that Microsoft seems to feel is essential for its help systems, to the much more useful aforementioned progress bars. Hopefully, future designers will be able to sort out the two types.

The sense of control, however, deserves more comment. While it is certainly true that a user's sense of control is critical for their acceptance of innovative technology, mismatches between their sense of and reality of control must be avoided. Confusion in pilots' understanding of the various flight control modes of commercial aircrafts have led to a number of crashes due to divergence of the pilot's sense of and their actual control authority. The automation has led them to feel they were in control and could accelerate normally when, in fact, their current control mode precluded the expected acceleration. The aircraft could "think" it was preparing to land, for example, and therefore the pilot's pushing the throttles forward would not produce the acceleration expected.

The final recommendation about allocating blame, apparently added by Norman himself, raises the interesting question of how to assign responsibility when there is some failure during use of highly automated systems. As Norman notes, it is still altogether too easy to blame a failure on the human operator when in fact the root cause of the problem is in the design of the automation. Automation in the aircraft cockpit is among

the most advanced, but misunderstanding of its operation nevertheless has been identified as a contributory factor to a number of crashes. In these cases, however, the primary cause of the crash was generally still identified as human error. Notably in one of them, an Airbus A330 crash at Toulouse on June 30, 1994, the pilot was the Chief Test Pilot for the aircraft [3]! It would seem that if even he could be led into a trap by the automation when operating in an unusual situation, what hope would the average pilot have? Apparently, the general public and legal view of the automation and its designer is not yet that of a completely responsible agent. So we cannot blame the automation . . . yet. Perhaps these views will change as information technology continues to develop.

One additional aspect of user interaction with automation not emphasized in the book is that the "smarts" of smart technology are based on not so much on information processing as on the creation and use of new sensors. Most of the illustrations of "smart" future systems such as appliances that toast your bread, brew your coffee, or otherwise take care of your house or drive your car, fundamentally depend upon the automatic systems having reliable access to new sources of information. Probably the most revolutionary is the availability via GPS of a user's precise geographic position. The inventor and futurist Ray Kurzweil [4] recognized this key role of sensors in his *American Scientist* article. In essence he argued: AI = I/O! This pseudoequation recognizes that the apparent intelligence of future automation will continue to be driven by access to new sources or information via development of new interactive sensors and actuators.

A final issue addressed only indirectly by Norman is the usefulness of what could be called an expressive interface. Such interfaces are not necessarily easy to use initially but with training can become extraordinarily powerful. Their development is connected with the especially difficult problem Norman identifies of transitioning a user from one level of automation to another and with the need for rich system feedback. A musical instrument such as a violin is a good example of an expressive interface. It is practically useless for music making to a novice because of the many degrees of freedom that need to be controlled. But once a user has learned to conform to all the constraints and can exploit all the dimensions of control offered by the instrument, an enormous range of sound may be produced and manipulated. Some of these same sounds could be produced by a keyboard programmed to play synthesized violin tones, but the user of the keyboard would not have anything like the degree of tonal control of an actual violinist.

An expressive interface like a violin takes time to learn, but it's worth it because the ultimate result is a far more intimate and faster connection between the operator and the system. The one feature about a violin that is simultaneously its strength and weakness is that it does not change much over time. Stasis is good since it provides fixed goals that can incrementally be achieved. It is bad because it is not adaptive. A better system would be an adaptive, expressive interface that somehow understands the expertise of its user and can adjust its displays and controls to provide just enough challenge to stimulate learning, but not so much to discourage use. But regardless of the amount of adaptation, expressive interfaces themselves become more difficult to use as their expressiveness, richness, and detail of their interaction with their users increases. They consequently discourage new users. Norman's comparison of the original Newton handwriting recognition system with that subsequently used on the PalmPilot identifies this need for system errors to be understandable. Comprehensible errors encourage users to continue interacting with a system and thus increasing the likelihood that they will be able to adapt to it. The PalmPilot recognition system provides an example of a successful, expressive interface, initially somewhat harder to use than the alternative keyboard entry system, but which provided significant long term benefits.

In summation it is important to note that Norman's book is somewhat deceptive. On the surface it appears to be a competent, journalistic account of recent and future technological developments, but, in fact, it is a meticulously, referenced analysis quite suitable as an introduction to a college-level course on automation. The author is not a reporter but a participant in the technology scene. Attentive reading, study and understanding of all the carefully picked references, suggested readings and cited books could easily amount to a minor in user–system interaction. Indeed, engineers currently designing future user interfaces to automated systems could do a lot worse than paying close attention to the recommendations, reasoning and references in this book. This book would be a great place to start and could be followed by Tom Sheridan's *Humans and Automation* [6], which contains very useful, more technical appendices. In fact, Norman's book could play a role similar to that of Fredrick Brooks' *Mythical Man-Month* [1], which for years has been supplementary reading for many programming courses.

I was only given a soft prepublication copy of Norman's book to prepare this review. Since it was after all a book about the future, I resolved to keep all my interaction online and to work only with soft copies. I did not print it out. This decision was a first for me since I am of a generation that still needs to look at a hard copy for critical reading. But despite the many great annotation features of Acrobat, I regret to report that it is still harder for me to think analytically about content on a screen. Information on a screen still seems to encourage reactions rather than reflection. Also, all the various great commenting tools for the soft copy accessed by clicks, pops and drags, intrude and disturb my thinking process. So for me at least, the user interface to word processors still needs work. It remains a fertile area for the coming designers of future things.

References

[1] F.P. Brooks, *The Mythical Man-Month*, Addison-Wesley, New York, 1975.

[2] B.G. Buchanan and E.H. Shortliffe (eds), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming*, Part 6, Addison-Wesley, Reading, MA, 1984.

[3] A. Degani, *Taming Hal*, Chapter 15, Palgrave-Macmillan, New York, 2004.

[4] R. Kruzweil, What is artificial intelligence anyway?, *American Scientist* **73**(3) (1985), 258–264.

[5] D.A. Norman, *The Design of Everyday Things*, MIT Press, Cambridge, MA, 1998 (also published under the title *The Psychology of Everyday Things*).

[6] T.B. Sheridan, *Humans and Automation*, Wiley, New York, 2002.

[7] B. Tognazzini, ASKTOG, 2008; http://www.asktog.com/basics/firstPrinciples.html.