Jef Raskin

Looking for a Humane Interface: Will Computers Ever Become Easy to Use?

the science of future technology

THE YEAR 1997 MARKS A HALF-CENTURY OF ELECTRONIC COMPUTING, BUT it also falls on a lot of decades and half-decades in the history of HCI (Human-Computer Interaction), personal computers, and even my own involvement in the field.

Landmarks in Human-Computer Interaction

Thirty-five years ago Ivan Sutherland was working on Sketchpad, it has been 30 years since I published my belief that computers should be graphics-based and human-oriented (as opposed to character-based and technically-oriented) [7], 25 years since Xerox Palo Alto Research Center was established, 20 years since Apple was incorporated, 15 years since Windows development was started [4], and 10 years since the Macintosh began to meet its sales goals and the computer world at large became aware of the importance of GUIs (Graphical User Interfaces).[1]

It is now evident that one of the most remarkable changes in the computing milieu was in neither hardware nor software per se but was embodied in an interface design concept that has since been implemented, with variations, on a variety of platforms and operating systems. The primacy of the interface was a revolution largely unforeseen by even the boldest of science fiction writers. Interface technology has altered, perhaps forever, what it means to use a computer, and is as significant for the majority of computer users as was the advent of low-cost computing itself.

The GUI, originally introduced primarily to provide a visual metaphor for an operating system, has changed our culture, sometimes in surprising ways. For example, when I first discussed "fonts" in connection with computers in the late 1960s most computer users didn't know what I was talking about. Fonts were the province of the typographer, the printer, and the graphic artist. Now even schoolchildren have hundreds of fonts to use. The encapsulation of access to the Internet in a GUI was certainly a major factor in its recent emergence as a fundamental resource of the computer age.

The landscape has changed: thirty years ago you could only have used a mouse at Doug Engelbart's group at SRI. Two decades ago it was only at PARC that you could find what the majority of today's users would recognize as a computer interface. But the Macintosh of a decade past would seem familiar to all of us.

[1] I proposed the Macintosh project to Apple's management in the spring of 1979; it was approved later that year.

Landmines in Human-Computer Interaction

We must not confuse the fact that our interfaces are far better than in pre-GUI days with any idea that they are optimal. The litany of complaints about computers is endless. Rarely does a week go by without one or more comic strips in our Sunday paper poking fun at the foibles of computers. Most of these complaints are due to failures of interface design, a topic which should be taken broadly to include everything the user must do from opening the box to attaching cables to installing software to how you shut down the machine (it's absurd to have to wait a few minutes while a computer prepares itself for you to turn it off, yet Windows 95 does exactly that).

Our present systems have come to be as large, complex, and nightmarish as the mainframes they first displaced (mainframes have become larger still; but most computer users don't have to deal directly with them on a daily basis). Today's personal computers are certainly more approachable than what came before them, but the GUI paradigm has not scaled well. We still cling to archaic named-file-based storage paradigms! To be a "power user" (one who uses a computer near its potential and can fix most problems single-handedly) you are expected to know intimately a mountain of information, such as the over 300 different settings or "preferences" of the system that sits before me. Today, if you can't fix a problem yourself, you may be faced with expense, loss of time, frustration, embarrassment, and at very least an unpleasant tryst with a slow voice-mail telephone system: "All our technical assistants are busy right now, but your call is important to us, and will be answered in the order received…". How often, when we find the problem, it turns out to merely be that one of those hundreds of choices was set incorrectly!

Most users end up treating all this detail as unknowable wizardry (in accord with Arthur C. Clarke's observation that any technology sufficiently advanced is indistinguishable from magic) and avoid "touching" things in hopes of keeping their systems stable long enough to get some work done. Some learn to avoid the computer itself whenever possible.

There is a normal human desire for a reasonably constant environment. An update is an upset. In this regard the current trend of releasing software incrementally makes it easier to managing a programming project. The potential for harm to users is incalculable. The same is true of interface toolsets; they encourage the rapid production of third-rate interfaces—often by unqualified personnel.

Lost on the Desktop

But the problems of today's GUIs run deeper than their hundreds of local design errors and ever-increasing complexity. The basic concept of operating-system-and-applications, even when the operating system is disguised as a desktop, is flawed from a cognitive perspective.

To make the discussion more concrete, let's look at a particular problem in as much depth as this short essay permits. Experimental psychology has shown that we can pay attention (be conscious of) only one thing at any given moment (the exceptions are few and specific enough to ignore in most interface design). If, as often happens, we do more than one thing at a time, all but one of them must be "automatic," that is, habitual. We are conscious of only one of them. It is as if our minds consist of a purely serial facility that is aware of a single stream of events and a multiple set of facilities that can run many tasks in parallel unconsciously, along with a process that shuttles tasks between the facilities [1]. Unfortunately, the design of present-day interfaces demands that we must be consciously aware of both the task at hand and the current system state. We cannot always safely surrender operation of the computer to our automatic, trained responses and so experience unnecessary frustration.

At present, each application in a GUI is a walled city with its own customs and habits. Some current systems allow a region with the behavior of application B to be embedded in a region with the behavior of application A. But this is only a patch on the problem—for when you are in a region, the behavior is determined by the local application. The way that spelling checking, for example, operates may suddenly change as you move from one part of a document to another. Your habitual modus operandi, learned in application A, may trip you up when you are in application B. Since your attention is on the content of what you are trying to do rather than system state, you may attempt to use the methods of application A. This will fail, and your attention will be displaced from your work to the interface itself, where you will resolve the problem. Then you have to refocus yourself on your work, having lost time and, often, your train of thought. Mode errors are a consequence of the human's single locus of attention.

A central goal of interface design is to allow users to make their own task the exclusive locus of their attention, by designing the interface such that it can be reduced to habitual operation. Too much of the emphasis of current GUI design, due to the marketing needs of past years, is ease of learning at the

expense of productivity (the explosion of "features" in the absence of global redesign has been widely discussed and is another major problem). As a result we have systems that are a combination of easy-to-learn menu-driven structures and relatively quick-to-use but hard-to-remember sets of keyboard commands. The user who desires efficiency has eventually to learn both. But the union of two wrong systems does not make for a single, unified, correct one. It is a common myth that an interface is either easy to learn or easy to use but not both. There is no theoretical reason to believe this, and there are counterexamples; unfortunately, there is not space here to discuss them.[2]

Measuring Interface Effectiveness: Cognetics

Too often I've been asked to fix a fundamentally failed interface by adding color and a fancy splash screen. It is like trying to repair a crumbling bridge with a clever paint scheme. Along with heuristics based on experienced, insight, and studies in psychology (the fine work of Don Norman and Ben Shneiderman comes to mind [5, 8]) we need to develop the discipline of cognetics, an engineering technology of applied physics, ergonomics, and cognitive psychology.

Some strides have been made in science-based, engineering-spirited HCI. Notable is the work based on the GOMS (Goals, Operators, Methods, Selection rules) model of Card, Moran, and Newell [2], and its subsequent development involving Critical Path Methods such as John's CPM GOMS [3]. Unknown to some and unused by most HCI practitioners (in my experience), this carefully validated work allows quantitative predictions of interface performance. I look forward to the combining of information theory and GOMS to define formal measures of productivity, for example by dividing the time to provide the minimum required input (derived from information theory analysis) by the time (as measured or as estimated by modeling) required by the system of computer and user for the task. We find, not surprisingly, that a dialog box that has no choices (e.g., you can only press ENTER before you can do any other task) has an productivity of 0. Desktop manipulations often have productivities under 0.1. Experimental work is needed to show to what extent such formal measures correspond to productivity in the field but the success of the GOMS model leads me to believe that the emphasis in interface design will increasingly shift from heuristics to cognetics [6]. Heuristics, if valid, will become consequences of the underlying theory.

GOMS is presently limited to expert use of systems; the future will see cognetic methods for the prediction of errors and the design and analysis of systems for novices. While fundamental insights and interface inventions will never "fall out" of the numbers any more than new theories can be summoned mechanically in physics, there is an imbalance at present in interface design: we lean too much in the direction of raw insight and guruism. Cognetics redresses the balance.

Beyond the Mouse

New input modes such as voice (foreseen in some technical detail by Vannevar Bush over 50 years ago), handwriting recognition, and direct mind-to-machine (MTM) links are less central to improving interface design than cognetic issues. While gloves and 3D input devices engender a lot of popular interest, the question of what are you going to have to do, say, write, or think in order to accomplish your goals generally goes unasked. I suspect most of us would prefer to use an MTM interface rather than type and shove a mouse around, but if the interface in which MTM is embedded is full of modal traps, complex navigational puzzles, and a multitude of details to be memorized, the improvement will be marginal and the interface will be as frustrating as anything now available.

There are a host of valuable new interface directions, such as spatial data representation or two-handed input devices, that have not been touched upon here in the interest of concentrating on more fundamental cognitive issues. Also note that GUIs make computers more difficult to use for the blind.

Financial, Ethical, and Political Issues

It is difficult to create a good interface where management does not think HCI is important. It costs money and takes time (in the short run). In my experience superior interfaces are exceptional long-term investments that:

- produce customer satisfaction,
- increase the perceived value of a product,
- minimize the cost of customer support,
- achieve a competitive advantage, and
- establish brand loyalty.

[2]The "CAT," a product of Canon in Japan (designed by the author) is such a counterexample.

The best interface can be undermined by poor implementation, system, or data design. If a system has bugs, crashes, allows incompatibilities, or does not facilitate data exchange between platforms and environments, it will not succeed.

Another factor would be helpful: in our legal system, medical personnel have some safeguards that allow them to practice their profession to high standards—for example, they can sue for wrongful termination if fired for refusing to follow a course of action that poses a threat to their patients. Structural engineers have similar protection from overzealous management. As an interface designer I have often been put in the position where I was directed to work within a set of design decisions that would increase the error rate, frustration, and hamper the productivity of users, thus doing psychological harm (my usual reaction is to quit: however this requires accepting a significant financial burden). Codes of ethics such as the ACM's, while helpful, do not serve to protect our livelihood. The future of interface design I'd like to see includes legislated protection for conscientious practitioners.

Can New Interfaces be Introduced?

Marketing a new and different interface, however much of an improvement it is, is seen as an insurmountable difficulty by most of the industry. This is a classic mistake: greatly improved products represent an opportunity. You cannot expect present users, especially in the business community which is all-too-aware of its huge investment in today's ways of doing things, to embrace a new system, but you can sell to legions of new users and millions of dissatisfied users. If the company providing the improved product has the stamina and financial resources to stay the course—and some luck—these new (and more productive) users will eventually come to the fore.

At present Silicon Valley and the other technological triangles, loops, and glens are somewhat complacent and consider the problem of interface design essentially "solved," needing only a bit of touch-up here and there. For many users and developers, today's GUI is the only conceivable interface. Their very success impedes the needed radical improvements. In spite of these formidable difficulties, we will do better. Those companies that cling to the status quo will not be able to also hold on to their customers. ◻

ACKNOWLEDGMENTS

Brian Howard, Linda Blum, Bill Buxton, Eric Slosser, and Dick Karpinski provided useful critiques and suggestions.

REFERENCES AND NOTES

1. Baars, B. *A Cognitive Theory of Consciousness*. Cambridge University Press, Cambridge, Mass., 1988. A view of the capabilities and limitations of human mental performance with strong implications for man-machine interface design.
2. Card, S., Moran, T., and Newall, A. *The Psychology of the Human-Computer Interface*. Erlbaum, Hillsdale, NJ, 1983. The first book on cognetics.
3. John, B. GOMS? *Interactions.* (Oct. 1995). A superb overview with pointers to the central literature. Her own work is worthy of careful study.
4. Linzmayer, O. *The Mac Bathroom Reader.* Sybex Inc., Alameda, CA, 1994. One of the more accurate histories of the Mac and very early Windows development, in spite of its title.
5. Norman, D. *The Psychology of Everyday Things.* Basic Books, New York, 1988. His most popular work is a delight to read. His technical work is of central importance.
6. Raskin, J. "Rationalizing information representation." A chapter in *Information Design*, R. Jacobson, Ed. MIT Press, in preparation.
7. Raskin, J. *The Quick Draw Graphics System*. Pennsylvania State University, State College, PA, 1967
8. Shneiderman, B. *Designing the User Interface.* Addison-Wesley, Reading, Mass., 1987. One among many of his books.

JEF RASKIN (jefraskin@aol.com)

It was my good fortune to have been at the vector that brought the infectious GUI into the personal computer industry. At first it was a concept too alien even for "visionaries." Without success I tried to explain the importance of graphical interfaces to Apple founders Steve Wozniak and Steve Jobs in the now-famous garage in 1976. I became their 31st employee in 1978 and the next year started a project to build a computer that I named "Macintosh" after my favorite apple, the McIntosh. At that time even Apple's contemporaneous Lisa project was a character-generator-based system, though it soon adopted the graphics-only display architecture from the Mac. It is little known that the Macintosh project was canceled by Jobs, and I had to continue it unofficially until I could get it re-funded. To get his support, I helped maneuver Jobs (who was fond of saying that innovation could not come out of large companies) to visit Xerox PARC and see a GUI in action. It worked. However, I got more than his support—in 1982 Jobs, having been ejected from the Lisa group,* took over the Macintosh project and I was offered the position of managing the Macintosh manual writing team. Having already run publications at Apple for a number of years, I left the company.

*See Stross, R. *Steve Jobs and the NeXT Big Thing.* Atheneum, New York, 1993. An incredible true tale of massive managerial mayhem.