Image Cover Sheet

CLASSIFICATION	SYSTEM NUMBER
UNCLASSIFIED	518635

TITLE

A Design Manifesto: Rethinking the User Interface

System Number:

Patron Number:

Requester:

Notes:

DSIS Use only:

Deliver to: CL

THIS PAGE IS LEFT BLANK

THIS PAGE IS LEFT BLANK

BEHAVIOUR & INFORMATION TECHNOLOGY, 2002, VOL 21, NO 3, 231–233

SL 2001-162

Taylor & Francis
Taylor & Francis Group

Book review

The Humane Interface: New Directions for Designing Interactive Systems By JEF RASKIN (Addison-Wesley, Boston, 2000) ISBN 0-201-37937-6 US $24.95 (paperback)

Review by Justin G Hollands, Defence R & D Canada, Toronto, Ontario, Canada

Jef Raskin is a user interface consultant and was a designer of the Apple Macintosh and Canon Cat computers. His recent book, *The Humane Interface* is of broad importance for the discipline of human-computer interaction (HCI) It is important for designers because it is one of the few books that offers an approach based on whatever science there is in HCI and, therefore, should offer user interface techniques that really work It is important for researchers because it helps us think beyond current interface styles, and makes us realize that a lot of what we take for granted with interfaces is at best unnecessary and at worst an impediment to our productivity.

It was a bit difficult at the outset to see the general picture Raskin would paint While reading the first few chapters, I found myself thinking I was getting an ersatz 'Design of Everyday Things' (Norman 1988) However, as I proceeded through the book, the various ideas were brought together into an overarching theme that, once understood, was profound This book is much meatier than Norman's, and I believe it forms an important new paradigm for interface design

Let us discuss some examples One of Raskin's ideas is that we use the same core set of elementary operations for many purposes (one might say across applications). Such operations include selection, indication, activation, moving and copying. If we use the same small set of operations across applications, we can learn a keyboard shortcut or have special purpose keys for each, rather than look them up via menu selection using a pointing device. Why do we have all these special-purpose menu items peppering the top of our displays? Raskin argues that they are unnecessary. In a mistaken attempt to provide us with lots of functionality (feature bloat), designers have cobbled together the elementary operations as menu items. But if the elementary operations are well learned, then a rapid series of command keystrokes will be faster than the menu look up and selection

Perhaps we do not need applications at all This revolutionary idea is explored at length in the book In Raskin's world, instead of buying software applications, we buy application-independent command sets that plug into our general user interface The command sets would work with transformers that change content from one data type to another (allowing you to check spelling in a graphic using optical character recognition, for example). Commands and transformers can be purchased as needed, perhaps even on an individual basis. The result is that we use a small set of elementary keystrokes or special keys for everything we do on the computer—word processing, e-mail, tables and graphs, computation, drawing pictures, writing code, etc. Because commands are consistently mapped to specific actions or gestures, software will be easy to learn and there will be no negative transfer from one situation to the next.

Raskin emphasizes the importance of automaticity in interface design If the same set of actions is repeated over time, it becomes automatized. He notes that there is a need for software that encourages habit formation. Such software would assign one control to one function mapping With special purpose keys, there is no problem in learning the association of a specific keystroke with a specific operation. In contrast, too many options puts the user's locus of attention to the selection of a command rather than the command itself. User says 'I think I'll bold this word here Should I bold this text from the <Format, Font> menu sequence, or by pressing the bold icon button? Or should I right click and choose font on that dialog box? Hey—that dialog box is different from the one I get with the menus!' The selection of the particular command sequence has become the locus of attention and distracts from the command action.

The flipside of this notion is that when we do develop automaticity we had better ensure that the sequence that is being developed is efficient Once the sequence is automatized it becomes hard to interrupt the sequence without attending to it Hence, if a designer prompts the

Behaviour & Information Technology
ISSN 0144-929X print ISSN 1362-3001 online © 2002 Taylor & Francis Ltd
http www tandf co uk journals
DOI 10 1080 01449290210155296

user 'Are you sure? Y/N' and the user nearly always responds 'Y', the user will incorporate the 'Y' keystroke into the command, and soon the action sequence for 'SAVE' is 'SAVE-Y'. Unfortunately, this defeats the whole purpose of the prompt, and adds a useless keystroke to the interaction. Raskin argues that it is better to provide an undo command so that if the user saves the most recent file erroneously, he/she can recover the old file easily He goes further to call the undo command 'an essential feature of any humane interface' (p 107) It is, unfortunately, an essential feature too often ignored

Raskin's emphasis on reducing keystroke count reflects a general emphasis on theory and modelling, including GOMS (Goals, Operators, Methods and Selection rules; Card *et al* 1983), Hick's law (Hick 1952), Fitts' law (Fitts 1954), and information theory (Shannon and Weaver 1949) Such topics are commonly neglected in HCI books. The utility of these approaches for design is nicely demonstrated through example and these would make excellent classroom or workshop exercises. I suspect that most interface designers do not make much use of such techniques and take a more instinctive approach. The state of interface design could be improved through the addition of these relatively simple modelling tools In the examples, GOMS is used to show the advantage of using keys rather than pointing devices. Hick's law is used to illustrate why it is better to use fewer menus with more items than the converse. Fitts' law is used to defend the idea of placing buttons at screen edge, and to argue for larger buttons Information theory is used to measure the efficiency of an interface, essentially by dividing the minimum amount of information necessary to perform a task by the amount of information that has to be supplied by the user The technique is illustrated nicely by example. Information theory is also used to evaluate interface elements such as radio buttons and toggles I particularly liked the assessment of one-button dialog boxes ('OK') as transmitting zero bits of information.

Raskin's treatment of modes (same command, different gesture in different contexts) is fundamentally important, because it makes clear that deviations in command structure across applications create modes (another reason for eliminating the application concept). 'For an interface as a whole to be classified as modal, it must not be modal for any gesture' (p 42) Raskin uses information theory to formalize the mode concept, thereby providing a technique for measuring how modal a particular interface is The mode concept is also illustrated when Raskin discusses transparent error messages that float over text, allowing one to see important text hidden by conventional error message boxes; it is still the case that some modern software prohibits you from selecting information in one window while another is open, producing a mode

Raskin talks of the difficulty in conventional navigation schemes, and proposes a general user interface for navigation he calls ZoomWorld ZoomWorld is conceptually similar to the walls of a planning room, where sheets of paper, sticky notes, photos, etc , are placed and organized on a large surface Because we tend to remember landmarks and relative position well, this type of organization should be effective ZoomWorld provides techniques for zooming in to achieve greater detail and Raskin has developed a prototype Note how this general interface would be used to access any information, eliminating the need for applications or modes I suspect however, that for the technique to be really effective, a large display surface would be necessary

Raskin talks at length about applications of the locus of attention concept. He takes the position that the user can maintain only a single locus of attention and there are time penalties associated with the switch So, for example, if a user is required to select a particular interface method for accomplishing a task, he/she will be distracted from the main task and its performance will suffer. If a user is in a stressful situation, the locus of attention narrows, making it more difficult for a user to diagnose a problem Errors result All the more reason for allowing the user to undo

The locus of attention concept is also illustrated nicely with the problem of interrupted work. We have all had the experience of returning to our workplace and having a document on the desktop remind us of a task we must complete Similarly, Raskin argues that software should be designed to return the user to where they last were However, most current software does not do this For example, when opening a Web browser, the user is shown a home page rather than that most recently visited. In current desktop systems, you must navigate to the task after boot up With Windows, if you keep files and applications open when shutting down, they generally do not open when the computer is booted. Reminders of incomplete work are lost.

And how about that annoying wait when you boot up? Raskin maintains that this problem would not be that difficult to fix technologically—it is just not considered a serious enough problem by computer manufacturers. I maintain that a simpler (but not as complete) solution would be to have the computer take all user input (e g. login, userid, any other prompts) at the start of the boot Then the user could go away and do something else (e g get coffee) while the computer booted As it stands, user input is often required at one or two different points during the boot up procedure,

about halfway through so that the user has to sit and monitor the whole dreary process

Raskin advocates for noun-verb command construction over verb-noun construction When bolding text for example, choosing the text and then bolding it is preferred over choosing to bold and then selecting the text This example makes clear how the latter construction sets up a mode. In the book a real-world example using product requisition illustrates the point nicely Raskin also advocates a text-search facility called LEAP. Searching for file names is highly restrictive and naming a file adds to the mental burden of the user Raskin takes the unorthodox view that all the text in a document is the file name, and all should be searchable (whole-text search) LEAP is accessed by a special purpose key (no mode problem), and is accessible in any context (again avoiding mode problems)

The book's material on cognitive psychology is dated and a little bit simplistic, but probably sufficient for interface design Raskin needs to remember that cognitive psychology is an evolving discipline, and our knowledge (such as it is) is not hard and fast. Reference to current topics in attention (e g executive control, task switching, spatial vs object-based attention, attentional capture) might have served Raskin well. For example, recent work in spatial attention suggests that it is possible to attend to two spatial locations simultaneously (Hahn and Kramer 1998, Bichot *et al* 1999) This result would have implications for Raskin's approach.

One could criticize this book for a lack of realism—are we really going to convince software companies to adopt consistent operations and stop making applications and start making special purpose commands and transformers? One might also criticize the notion that the set of all possible commands can be reduced to a set of a few core operations Can we really classify all possible commands into just a few core operations? Would we be restricting ourselves in any way by doing so? However, these are the types of criticisms that can be aimed at all great ideas—too idealistic, too simplistic, too 'out there' Personally, I would rather see idealistic, simplistic and 'out there', than jaded, complex and pedestrian The core notion of application as mode and the difficulties that result, and the design solutions based on locus of attention and automaticity are of fundamental importance Raskin must be given credit for taking a brave stand that is both deep and wide reaching If you only have time to read one HCI book this year, make it this one

References

Bichot, N P, Cave, K R and Pashler, H 1999, Visual selection mediated by location Feature-based selection of non-contiguous locations *Perception & Psychophysics*, **61**, 403–423

Card, S K, Moran, T P and Newell, A 1983, *The psychology of human-computer interaction* (Hillsdale, NJ Erlbaum)

Fitts, P M 1954, The information capacity of the human motor system in controlling the amplitude of movement, *Journal of Experimental Psychology*, **48**, 483–491

Hahn, S and Kramer, A F 1998, Further evidence for the division of attention among non-contiguous locations, *Visual Cognition*, **5**, 217–256

Hick, W E 1952, On the rate of gain of information, *Quarterly Journal of Experimental Psychology*, **4**, 11–26

Norman, D A 1988, *The design of everyday things* (New York Doubleday)

Shannon, C E and Weaver, W 1949, *The mathematical theory of communication* (Urbana, IL University of Illinois Press)