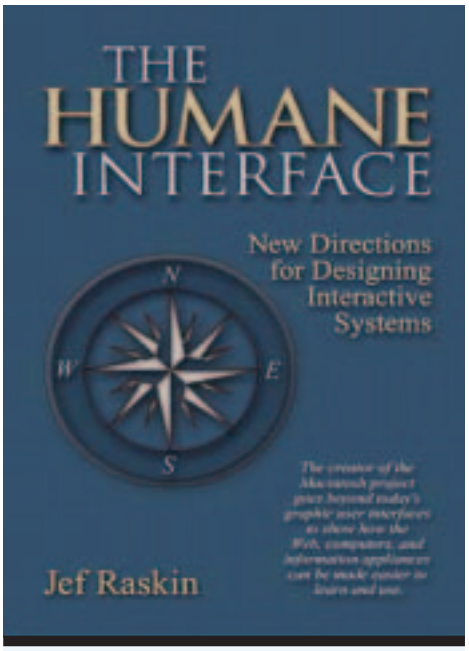
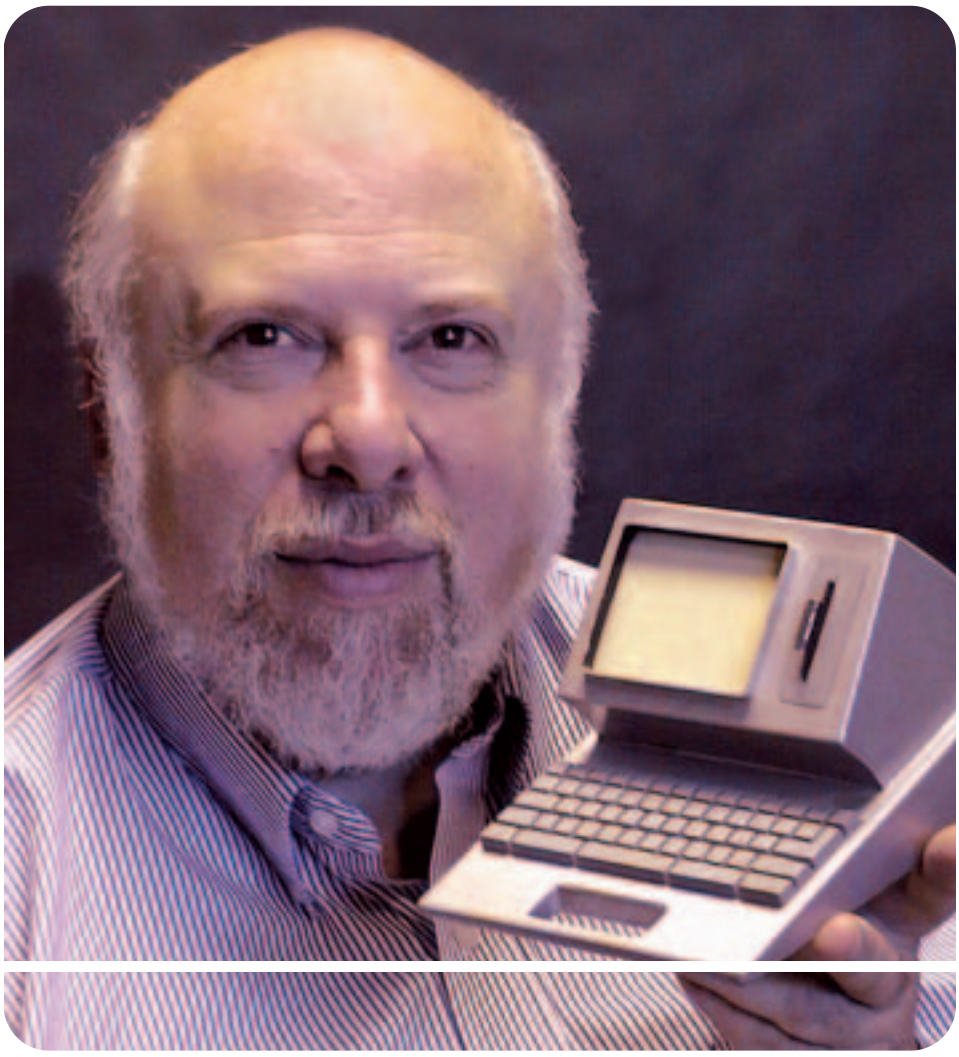
From Apple to Archy: The Visionary Journey of Jef Raskin



The Humane Interface
By Jef Raskin
Addison Wesley, 2000
Reviewed by Charlie Kreitzberg



Jef Raskin, who headed the Macintosh *project from 1979 to 1982, died on February 26, 2005, at age 61. Like many of the pioneers in personal computing, Raskin was a Renaissance man—a musician, composer, and conductor. His sculptures have been exhibited at New York's Museum of Modern Art; one is included in the permanent collection.*

The fog that surrounds corporate history has created several versions of the origins of the Macintosh and Raskin's role in it. According to Arik Hesseldahl's 2005 Forbes article, Raskin wrote his 1967 master's thesis on GUIs and then encountered the Xerox Star at the Palo Alto Research Center in the 1970s. After visiting the garage in which Jobs and Wozniak were building their upstart company, Raskin became Apple's thirty-first employee in 1977.

In 1979, Raskin proposed building an easy-to-use computer for consumers. According to Wired News, Raskin recalled, "There was this thing that I'd been dreaming of for some time, which I called Macintosh. The biggest thing about it was that it would be designed from a human-factors perspective, which at that time was totally incomprehensible."

It was not an easy sell. "[Steve] Jobs hated the idea," Raskin said. "He ran around saying 'No! No! It'll never work.' He was one of the Macintosh's harshest critics, and he was always putting it down at board meetings. When he became convinced that it would work and that it would be an exciting new product, he started to take over."

To honor Jef Raskin's contributions to the field of interface design, User Experience *is publishing this review of his book,* The Humane Interface: New Directions for Designing Interactive Systems.

Jef Raskin spent a large part of his life passionately thinking about computers and how best to interact with them. He summarized a lot of that thinking in his book *The Humane Interface*. *The Humane Interface* is not a how-

to book. Rather, it is a sweeping look at the field of human interface design, and a foundation for Raskin's vision of a new paradigm for human-computer interaction. Like Alan Cooper's books, it is a mixture of ideas and opinion; those seeking a nuts-and-bolts approach to interaction design may be disappointed, and those who have been in the field for many years may find some of his discussions of the fundamentals rather simplistic. But for those who are trying to understand the field in holistic terms, and those who want to take human-computer interaction to the next stage, this book is essential reading.

For those who are trying to understand the field in holistic terms,... this book is essential reading.

Instructors who are looking for material to help students get a conceptual grasp of the field may also find this an excellent text, particularly if supplemented by a comprehensive review of the literature such as the one provided by Shneiderman and Plaisant's *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (4th Edition), also published by Addison Wesley.

On first reading, I found *The Humane Design* a bit puzzling. The initial chapters focus on presentation of basic cognitive principles of UI and a well-written recap of the Goals, Operators, Methods, and Selection Rules (GOMS) model, Fitt's Law, and Hick's Law. These are useful for the less experienced reader but seem neither particularly compelling nor profound. In reading on, however, it becomes clear that Raskin is laying a foundation for a rethinking of the basic UI paradigm that has been with us since the days of the Xerox Star, Lisa, and Macintosh. This new paradigm was originally called *THE* (for The Humane Environment) and later renamed *Archy* (a play on R-CHI, the initials of the Raskin Center for Humane Interfaces).

Foundations

Raskin felt that the first step in creating an effective UI paradigm is to ensure that it fits comfortably within the human cognitive processes. He identifies a key characteristic of a good UI paradigm as "automaticity"—the ability of the user to rapidly habituate to the UI so that interactions become effortless.

To achieve automaticity, the UI must align with the underlying cognitive process. Accordingly, he begins with a discussion of cognitive foundations. In the chapter "Cognetics and the Locus of Attention," Raskin explores the relationship between interaction design and attention. While attention is one of the key cognitive processes that UI designers must consider, Raskin does not discuss closely related concepts such as recognition vs. recall, perceptual grouping, concept formation, and mental models of process, which are also part of the underlying cognitive foundation of UI design.

As a musician, Raskin was aware that musical notation is, at best, a fair approximation of the composition to be played—a situation that is even worse in dance choreography. Perhaps this is the reason that he begins with a discussion of the problems of creating precise specifications for sequences of actions for keystrokes and mouse movements and presents a workable, if somewhat cumbersome, notation for accomplishing this task.

UI a la Mode

Raskin held that UI interaction should be modeless. A mode is a state in which specific actions take on unique meanings. For example, dragging the mouse in a paint program can have different effects depending on what tool is selected. In Microsoft Word™, pressing the letter keys may have different consequences depending upon whether you are in Insert or Overwrite mode.

Raskin's view is that modes add unnecessary complexity by increasing cognitive load on users, who must remember what mode is currently active and deal with the overhead of switching from mode to mode. Raskin suggests that designers employ "user-maintained"

modes in which the user actively maintains the mode while the operation is being completed. An example of a user-maintained mode is typing while holding down the shift key. Contrast that with the alternative of pressing the Caps Lock key, which creates a pure mode. Before I purchased a keyboard that allowed me to disable Caps Lock, I actually pried the offending key off my keyboard because of the frequency with which I pressed it by mistake.

Raskin also argues for the superiority of noun-verb interaction over verb-noun constructions. For example, it is generally easier to select text and then press the delete key than to press the delete key and then indicate what text should be deleted.

Raskin is skeptical of the value of customization and of interfaces that attempt to cater to both novices and experts. I do not agree with this position. It seems to me that customization, although usually poorly implemented, is important in dealing with individual differences. We know there are cognitive stylistic differences among individuals, as well as differences in the types of tasks and processes that bring users to a particular application. A customizable interface can be extremely helpful in helping a software application fit real-world needs.

Abolishing Applications

Raskin would probably have replied that there is no need for software applications. He envisions a UI in which applications disappear. He points out that, "the present structure of computer software, consisting of an operating system under which application programs operate, is inherently modal," (p.139) and argues that, to create a non-modal system, applications should be abolished.

In other words, the only thing that makes operations such as "save the document" consistent from application to application is that most UI designers adhere to the same standards. Raskin wants all such basic operations to be an integral part of the operating environment.

Less basic operations could be purchased from vendors, he suggests, as application commands that can be applied to content. So you would purchase a spell-check command as part of a collection of editing commands rather than buying a word-processor application. This is reminiscent of how programmers view object libraries. Some of these commands would be "transformers" that perform data conversions when needed. For example, if you attempted to apply the spell-check command to a photograph, the command would invoke a transformer that would perform an OCR operation on the photo to extract any textual information in the image. This transformed data could then be acted upon by the spell-check operator.

For navigation, Raskin envisioned a paradigm based on panning and zooming. Imagine viewing the World Wide Web from outer space, then zooming in on your company, your personal computer, a particular document, a paragraph, and finally, a word. Raskin called this metaphor "ZoomWorld"; a demonstration is available at www.raskincenter.org.

Conclusion

Raskin believed that current GUI approaches are bloated, outdated, and inconsistent and should be replaced by a new paradigm. He began the design of a new paradigm with the ill-fated Canon CAT computer. Unfortunately, this computer did not have the commercial success that would have exposed his ideas to a wider audience.

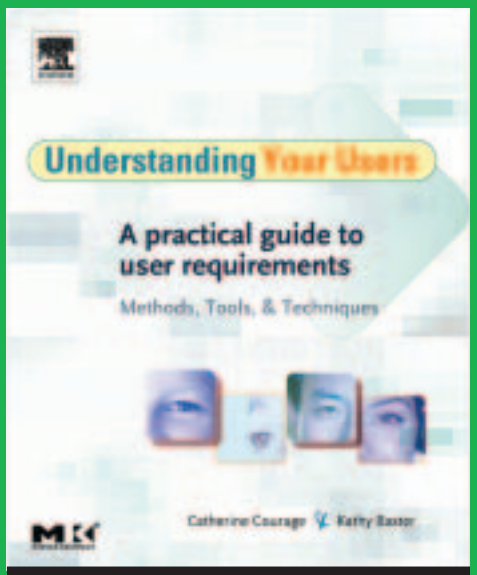
To continue his work, Raskin founded the Raskin Center for Humane Interfaces. Although his work is not complete, there are plans to release a prototype of the proposed interface paradigm (Archy) in the near future.

Undertaking to revise the basic interaction paradigm is a daunting task and it is difficult to see all the ramifications. Is Raskin's vision robust, efficient, intuitive, and extensible? I don't know. While I don't feel comfortable with all of Raskin's conclusions, his vision is vast and his arguments are thoughtful.

Asked in a 1999 interview (see www.mymac.com) about his most significant professional achievements, Raskin replied that, "convincing Apple that it was better interfaces and not better hardware that it needed to stay alive after the Apple II, and creating and leading the Macintosh project to implement that insight, are wonderful credentials to have in my résumé."

But for Raskin, the Macintosh was the past. The future was Archy. All of us in the usability profession have something to learn from him. It is not necessary to agree with all of his conclusions to hope that his vision will have a major impact on the future of HCI.

Everything You Ever Wanted to Know About Gathering User Requirements



Understanding Your Users: A Practical Guide to User Requirements
By Catherine Courage & Kathy Baxter
Morgan Kauffmann, 2004
704 pages
Reviewed by Charlie Kreitzberg

Your first thought when you pick up this impressive book is, "Wow, it's heavy." The second is, "It's beautifully produced." And when you begin reading, you will be impressed by the encyclopedic coverage and thoughtful presentation.

Understanding Your Users is a comprehensive 704-page volume that covers interviews, surveys, needs analysis, card sorting, task analysis, focus groups, and field studies. This is a pragmatic book, well organized for reference or learning. There is an abundance of tools and templates. Case studies and examples clarify the techniques presented.

The authors have paid a great deal of attention to the usability and organization of the book. One of the real strengths of this book is its level of detail; "lessons learned" and tips abound. The authors have tried to provide the reader with everything needed to execute a technique flawlessly. This will not only help the less experienced reader, but will help when you need to bring a programmer, manager, or business analyst up-to-speed quickly.

While you are unlikely to take it to the beach for summer reading, *Understanding Your Users* will be a great addition to your library. **UX**