PUBLICATION NOTES

Books Received for Review

CASE-BASED PLANNING: VIEWING PLANNING AS A MEMORY TASK

Kristian J. Hammond, Academic Press, 1989.

WORKING WITH COMPUTERS: THEORY VERSUS OUTCOME

Gerrit C. Van der Veer, Thomas R.G. Green, Jean-Michael Hoc, and Dianne M. Murray (Eds.), Academic Press, 1988.

EXPERT SYSTEMS FOR ENGINEERING DESIGN

Michael D. Rychener (Ed.), Academic Press, 1988.

NEURAL COMPUTING ARCHITECTURES: THE DESIGN OF BRAIN-LIKE MACHINES

Igor Aleksander (Ed.), MIT Press, 1989.

EXPERT SYSTEMS: AN INTRODUCTION FOR MANAGERS

Anna Hart, GP Publishing, 1988.

If you are interested in reviewing any books mentioned above or some other book, please contact me (Joe Sullivan), and I'll arrange for you to receive a review copy. If you have recently read a book that you think would be of interest to the CHI community, feel free to submit a review for publication. Also, if anyone comes across a good article or book that is not in the mainstream publishing channels, please let me know about it and I'll pass it along to everyone.

Books Sent Out For Review

PERCEPTRONS: AN INTRODUCTION TO COMPUTATIONAL GEOMETRY, Expanded Edition. Marvin Minsky and Seymour Papert, MIT Press, 1988.

Reviewer: Patricia J. Hoffman, Lockheed Missiles & Space Company, O/L2-90, B/572 1111 Lockheed Way Sunnyvale, CA 94089-3504

ADAPTIVE PATTERN RECOGNITION AND NEURAL NETWORKS Yoh-Han Pao, Addison-Wesley Publishing, 1989.

Reviewer: Charles F. Hall, Ph.D. Lockheed AI Center 2710 Sand Hill Road Menlo Park, CA 94025

BOOK REVIEWS

HUMAN-COMPUTER INTERACTION: A DESIGN GUIDE,

Mark K. Jones, Educational Technology Publications, 1989.

Reviewed by Steven Tripp, Instructional Technology Center, University of Kansas, Lawrence, KS 66045

When reading a book about designing human-computer interaction, it adds to the reader's confidence if the book is well-written. After all, if someone claims to know something about "user-friendliness," his book should reflect some of the same

kinds of considerations that go into software design. Happily, Mark Jones' *Human-Computer Interaction: A Design Guide* meets that criterion.

Jones' book is divided into four parts: an introduction, the user's model of the system, display design, and dialog design. The author states that the book is intended to be a guide for professional designers rather than a set of guidelines. That is, it is an overview of design issues rather than a set of prescriptions. The author is so concerned to make this point that he repeats it at least four times. Therein lies a dilemma. It is difficult for a book so short (about 100 pages) to cover the issues in sufficient depth to allow unsophisticated designers to make informed judgments. If one is in need of guidance, one is likely to fall back on the prescriptions rather than the rationale. Since individual applications are always unique, in the absence of experience one can only use the guidelines. Fortunately, the guidelines, which are summarized at the end of each section, are carefully chosen.

The first substantive part of the book deals with the various models of the system that may exist and the orientation of the user within the domain of the computer. How can the user tell what state the machine is in and how can he tell what moves are possible from that state? It is pointed out that the designer and the user are likely to have radically different **models** of the machine due to the designer's immensely greater knowledge of the system and its history. Therefore, great care must be taken that the user can form a coherent model of the machine without deep knowledge of its underlying structure.

One method of providing the user with a useful model is to use metaphors. A metaphor creates a relationship between a familiar thing and an unfamiliar one. If the metaphor is carefully chosen and exploited, it allows a novice to transfer old knowledge to a new situation. Of course, such metaphors as menus, ports, drivers, windows, and scrolling appear spontaneously as we expand the types of computer objects and events we have to deal with. However, the design of software intended to be used by novices requires consciously chosen metaphors which do not call upon esoteric knowledge in order to be interpreted meaningfully. Needless to say, the choice of good metaphors is not a process that can be done without creativity, and that makes the writing of guidelines a problem. As users move or browse through a system, they may become disoriented by a seeming lack of relationship between succeeding screens.

Aside from the crucial maxim that the model of the system be consistent, the second most important feature of software is that there be landmarks that may be used for orientation as users navigate through the system. One way of doing this is to utilize the space of the computer screen to spatially represent data . Another way is to use iconic symbols or distinctive visuals to orient the user.

In addition to the need to orient the user, there is also the problem of visual momentum, the cumulative effect of succeeding displays. This section of the book, covering a topic which has not been overly discussed, is especially worth reading. It must be noted, however, that the section on browsing anticipates but does not mention hypertext, a strange fact for a book published in 1989. Given the recent enthusiasm for hypertext-like applications, a software design guide would seem to need a section on this topic.

The portion on closure is interesting and informative. By making a distinction between lexical, syntactic, and semantic events, that is, relatively low-level activities like clicking a mouse and relatively high-level activities like opening a new document, it can be explained why some delays are more tolerable than others. In general, low-level activities require quick closure and therefore short delays. High-level activities do not require quick closure and therefore delays of many seconds may be tolerable. Whether short or long, delays should be followed by some discernable act of closure or users will feel dissatisfaction.
The discussion of the user's model of the system also covers such topics as progressive disclosure and modality, which are related to the problems of inconsistency and disorientation. It seems that if two prescriptions could be derived from the research, they would be that the designer should be concerned above all with providing the user with a consistent model of the system, and with providing the navigational tools needed to move efficiently through that system.

The next part of the book deals with the sticky problem of perception and display design. Although quite precise statements may be made about perception, those statements do not translate directly into designs. Many of the prescriptions in this section are in the form of warnings rather than suggestions. Don't use all uppercase letters. Don't use too many colors. Don't right justify without proportional spacing. Don't use pure blue for text. To some extent that advice in this section is a function of the machines we use and, thus may be outdated already. Reading rates, for example, are clearly dependent upon the the quality of the display screen. One can imagine that in the near future input and output devices will have greater variety and bandwidth, and therefore the associated design problems will be considerably different from those we face today. Indeed, Jones' discussion of lightpens, touchscreens, and mouses seems

outdated already. Can there be any software designers out there who really need an explanation of what a mouse is? Other aspects of display perception, such as screen angle and room lighting and glare, which are mentioned in the book, are beyond the control of the designer. All of which serves to underline the complexity of designing for perceptual ease and the problem of providing an overview that goes beyond guidelines.

The last section of the book is on dialog design. Jones begins with a discussion of theories of memory and the learning of command languages. Although there is mention of the need for lexical congruence in command names, there is nothing about the need for syntactic design. Mention of Payne and Green's (1986) "Task-Action Grammar" would have been useful to command language designers, because it allows a measure of learnability. Since users clearly have abstract representations of command languages, a technique for representing those abstractions should be part of a design guide.

Jones continues with a short discussion of the severe difficulties of using natural language interfaces, which leads to a contrast with the relative success of direct manipulation interfaces. Within this discussion Jones mentions the little-known technique of sketch recognition, a process by which hand-held device movements are interpreted and acted upon. Whether this technique will find acceptance remains to be seen. Jones concludes with short passages on menu and icon design (topics which, given their importance, deserve more coverage) and help file design. This section, like the others, ends with a list of prescriptions.

I conclude by recalling the dilemma mentioned at the beginning. Can a short book aimed at **the professional designer of human-computer interfaces** be more than a list of prescriptions? Perhaps it can, if it orients the designer to the types of issues that need to be considered and points towards the literature on those issues. More likely, I suspect most professional designers will either be familiar with the issues or, if not, will look to the lists of prescriptions for quick answers. On the other hand, this might be a good book for students, who are likely to lack both the breadth of knowledge and experience needed for the complex task of interface design.

Reference:

Payne, S. J., Green, T. R. G. (1986). Task-Action Grammars: A Model of the Mental Representation of Task Languages. *Human-Computer Interaction*, 2, 93-133.

COMPUTER-SUPPORTED COOPERATIVE WORK: A BOOK OF READINGS

Irene Greif (Ed.), Morgan Kaufmann, 1988.

Reviewed by Patricia J. Hoffman, Lockheed Missiles & Space Company, O/L2-90, B/572 1111 Lockheed Way Sunnyvale, CA 94089-3504

This book has selectively collected 28 Computer-Supported Cooperative Work (CSCW) papers from journals including the *IEEE Computer*, *Proceedings of the Conference on Human Factors in Computer Systems*, *Human Computer Interaction*, *American Psychologist*, *Harvard Business Review*, *Management Science*, *AI Magazine*, *Scientific American*, *Atlantic Monthly*, *TOOIS*, and *CACM*. From the list of publications, it can be seen that not only does CSCW borrow from varied fields, but that this text will be of value to CSCW for consolidating the information and also to the various fields with which it overlaps. The papers, ranging in date from 1945 to 1988 and written by many notable authors including Douglas C. Engelbart, Alphonse Chapanis, Thomas Malone, Jeff Conklin, Kevin Crowston, Sara Kiesler, and Terry Winograd, build upon each others' arguments with many of the latest papers referencing earlier works. The editor, Irene Greif, has included two of her own papers (coauthored with Sunil Sarin): "Computer-based Real-Time Conferencing Systems" and "Data Sharing in Groups Work." She also provides excellent commentary on how the papers relate to each other and build on the field of CSCW.

These papers start with a historical perspective and continue through design theories for CSCW. The following paragraphs review in more detail each corresponding section of the book.

Overview

The overview has succinct definitions of Computer-Supported Cooperative Work (CSCW) and groupware, relating them to ranges of disciplines: computer science, artificial intelligence, psychology, sociology, organizational theory, and anthropology; and differentiating them from computer science subfields: human factors in computing, computer-human interaction, office information systems, management information systems, and group decision support systems.

Part I: Visions and First Steps Towards Computer-Supported Cooperative Work

The first few papers give historical perspective to the emergence of CSCW as well as much of the rest of modern computer science. It also provides visions to open issues of CSCW research including shared-screen concurrence, role hierarchies for defining user capabilities, and social issues affecting the adoption of skills that allow new artifacts to become everyday tools.

The papers include early research projects both at Douglas Engelbart's laboratory at Stanford Research Institute and at Alphonse Chapanis's laboratory at