FOUNDATIONS OF HUMAN-COMPUTER INTERACTION

Brian R. Gaines and Mildred L. G. Shaw

Department of Computer Science, York University,
4700 Keele Street, Downsview, Ontario, Canada M3J 1P3

Gaines, B. R. and Shaw, M. L. G. (1985). Foundations of human-computer interaction. In Proceedings of CIPS/ACI CONGRESS 85, pp. 384-388. Canadian Information Processing Society, Montreal.

Abstract: This paper is concerned with techniques and principles for programming effective human-computer interaction. Interest in human-computer interaction has existed since the early days of computing, became of high significance in applications as interactive computing came into widespread use in the 1960s, and is fundamental to the fifth generation computing system development program. Human-computer interaction studies are in a transition to deeper theoretical models from which design principles can be derived. This paper demonstrates that the formal analysis possible for the computational aspects of software engineering can be extended to the human aspects also.

Dans cette dissertation, is s'agit des techniques et des principes pour la programmation d'interaction effective entre humain-ordinateur. L'explosion des ordinateurs dans les années soixantes a rendu la computation interactive significative en applications, et est fondementale au programme de dévelopement pour systèmes d'ordinateurs de cinquième génération. Les études d'interactions entre humain-ordinateur sont dans une transition d'approfondir les modèles théoretiques desquels les principes de plans peuvent être dérivés. Cette dissertation démontre que l'analyse formelle possible pour l'aspect de calcul en génie logiciel peut être étendu aussi aux aspects humains.

Introduction

The study of human-computer interaction (HCI) is undergoing a major transition. The 1960s was an era of practice and experience. The 1970s was an era of pragmatic design principles based on that experience. The 1980s will be seen as the era of theoretical foundations for HCI in which those design principles became first codified and then generated from deeper principles. It is difficult in the midst of a transition to see where one will arrive eventually. If the next plateau were obvious it would provide a goal to which we could step rapidly. We can see clearly where we have been. We can broadly characterize our goal. Each of us can attempt to crystallize out a more detailed model of that goal. Eventually some will succeed, the new plateau will be recognized and a concerted effort will rapidly substantiate the model to form the new reality.

Four Strategies in HCI Design

Norman [26] suggested four strategies for providing design principles for HCI which can be summarized as:

1: Raise awareness of system designers of significance of HCI;

2: Provide methods and guidelines for HCI design;

3: Provide software tools for HCI design;

4: Separate HCI design from other design tasks.

The first strategy has been an ongoing effort since the late 1950s. While there were early complaints that it was not working [24, 33], and later evidence of many systems still operating with poor HCI [25], the strategy is now highly effective. Hardware/software technologies have converged to almost uniform effectiveness and market forces have come to emphasize HCI as the main basis of product differentiation [37].

The second strategy has been an ongoing effort since the early 1970s [17, 21, 10, 20] but has been so effective in generating guidelines as to become ineffective in promoting their use. When over 500 rules for effective HCI are grouped together [36] they become daunting and the user begins to ask for them to be condensed or automated.

The third strategy is that apparent in the 1980s and has resulted in the provision of tools for HCI design such as Moran's [22] **command language grammar**, Jacob's [19] **executable specifications**, and Edmonds and Guest's [2] **user interface manager**. The tools take the load off the HCI designer and allow effort to be focused on generating effective HCI, but they do not in themselves provide a dialog style promoting uniformity and ease of transfer between systems.

The fourth strategy is a logical conjunction of the second and third, to provide tools that enforce a dialog style. This can be used to automate HCI guidelines through application-independent **dialog shells**. The Xerox Star user interface [31] was an early example of such a shell and recent "window" systems provide shells for personal computers [16]. INTELLECT [18] and ASK [32] are examples of natural language shells for database access. Such shells separate the user interaction from the application in so much as this is possible, but still leave open those aspects of the interface which are application-dependent, often the most significant part.

A Fifth Strategy

These four strategies capture the state-of-the-art in HCI practice, what can usefully be done now. For the future we suggest a fifth strategy that takes into account three major requirements in advancing the state of the art:

(a) We need HCI principles that capture **all** aspects of the roles of people in human-computer systems, not just those aspects of the interaction which are application-dependent;

(b) We need theories that will enable us to design systems, including a variety of shells, that are locally optimized for individual HCI and globally optimized for overall performance;

(c) Such theories must contain adequate models of people **and** computers, and of their interaction.

The fifth strategy is:

5: Provide an overall framework for HCI in systems involving multiple people in various roles and multiple computer sub-systems at different levels. From the overall framework derive the other four strategies: separation of HCI from application; software tools; design principles; and significance of HCI for total system performance.

It may seem unreasonable at the current state-of-the-art to attempt to adopt such a strategy when the development of the first four are not yet complete. However, there is ample evidence that the first four are widely accepted and becoming implemented in pragmatic commercial situations. We believe that the cutting edge of HCI research studies must now move to the provision of deep theories, guided and tested by the ongoing results of the first four strategies. We see a danger that too much effort will go into optimizing the interface to the detriment of overall systems. We also see the danger of the field fragmenting so that problems of the psychology of programming languages, for example, are treated quite separately from problems of user interaction. This would be out of step with trends in computer system design, such as those in knowledge-based systems, where programming and interaction are becoming inseperable user activities. Some specialization of HCI according to roles is necessary but it is desirable that this be derived from overall principles.

We emphasize the need for models of computer systems to be developed that are well-suited to the analysis of HCI. The computer and the person are both essential sub-systems of the HCI situation, but there is a tendency to assume that the missing formalization is only that for the person. This is not so. We are equally in need of formalizations of computer systems that can be used in conjunction with those of the person for overall system design. Models of both sub-systems are necessary that can be used together in a way that accounts naturally for their interaction, for the reflection of the user model necessary in the computer and for the reflection of the computer that is the user's model. We have previously emphasized the need to combine systemic, psychological and computational models in the design of dialog shells [13]. The need is even greater in considering total system design.

This paper provides a framework for HCI that subsumes the systemic, psychological and computational aspects of HCI.

The Virtual Machine Hierarchy

The concepts in this paper go back to a study of the nature of analog computing [4] in which a category-theoretic model of "virtual" machines was used to formalize what is "natural" in the relation of an analog computer to that which it models. This approach was later generalized to include people and analyse questions about the naturality of programming languages and user interfaces [5]. The model has since been used to analyse the relationship between hardware engineering and software engineering [8] and the role of the behavioral sciences in programming [9], and the category-theoretic model of analogy has also been developed [12].

Virtual Machines

Our model is based on the concept of **virtuality** which has proved an extremely useful one in computer science [5, 34]. We can view a system consisting of a computer together with a usable set of programs either a programmable machine with certain features together with a program, or as a (programmable) virtual machine (VM) with extended features. This encompasses simple virtuality such as virtual memory where accessing certain locations causes automatic paging, through subroutines regarded not as procedure calls but rather as instruction set extensions, to high-level languages and library systems that appear to their users to have totally different features to the machines on which they run.

VM's form a natural hierarchy from microcode through to applications. In Fig.1, for example, the natural language interface of SHRDLU is a program in PLANNER which is written in LISP which is written in an intermediate language, and so on. The extra arrows indicate that PLANNER could have been written in some

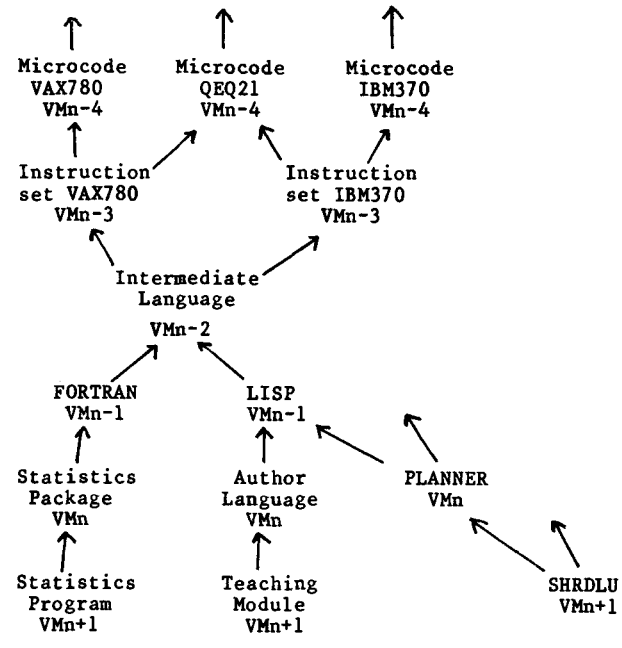

Fig. 1 A virtual machine hierarchy

language other than LISP and SHRDLU could have been in something other than PLANNER. The remainder of the figure shows relations between different levels of the VM hierarchy, and the role of, for example, an intermediate language in decoupling language translators from actual machines.

Technical Analysis of VM Relations

The technical relationships between VM levels, that VMn-1 provides operations suitable for encoding VMn, may be analysed formally in algebraic terms [4, 5, 12]. The requirement that VMn+1 emulates a particular process may be represented as the existence of a pair of functors forming a pullback from a "truth category", representing the minimal analogy required for emulation, to the two categories representing VMn+1 and the process it emulates. The semi-lattice of pullbacks under the truth category represents various analogies between VMn and the emulated process that contain additional but strictly unnecessary detail.

The naturality of the relationship between VM levels may be analysed in terms of computational complexity. If VMn retains all the functionality of VMn-1 but offers access to it in a different way then what has been gained? What is the difference between a high-level language and an assembler? In complexity-theoretic terms we can consider a class of problems together with a measure over that class and examine the expected length and expected run-time of a program for this class/measure on VMn-1 and on VMn. If the expected length and run-time are substantially less for one than the other then the first VM is better orientated to the class of problems. If one is substantially better and the other substantially worse then we have a trade-off between simplicity of program and efficiency of operation.

The Programmer in the Virtual Machine Hierarchy

These technical analyses miss some major issues, however. The value of an analogy between a system and its emulation must be viewed in terms of the cognitive value of that analogy to the programmer. The complexity-theoretic length of a program is not a useful measure if that program is not structured comprehensibly for the programmer, if it is not natural

for human generation. The de-coupling between levels of the VM hierarchy is itself important only in relation to a programmer who need be aware only of the level which he accesses. If, as in Fig. 2, the VM hierarchy is extended to show the people involved in its design, implementation and use, a surprisingly large number of HCI studies may be set in an integrated framework [9].

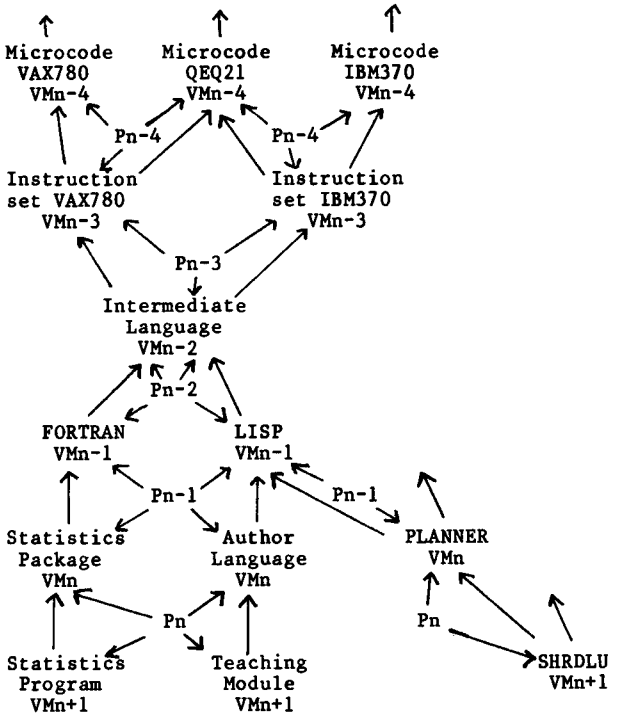

Fig. 2 VM hierarchy extended with programmers

Thus, in considering the implementation of a VM there is not just its technical relationship to the machine to be programmed at the next level, but also the relationship of these two machines to the programmer. Fig. 3 shows the kinds of questions that may be posed about these relationships and illustrates that an effective implementation involves positive answers to all three questions. The complexity-theoretic relation between VM levels places lower bounds on the length and run-time of programs, but these may be substantially exceeded if the programmer does not understand one or both of the VMs involved. It also enables us to define a problem-orientated language as one in which the problem analysis and the implementation specification are the same.

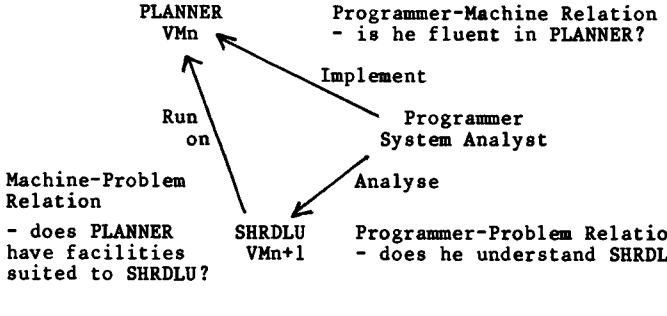

Fig. 3 Three relationships between machine, problem and programmer

We have remarked [5] that the algebraic analysis of the VM hierarchy might be extended to include the programmer if we had adequate category-theoretic models of the relevant cognitive system. Some of the formal models being developed in cognitive science show promise of providing this [35] such as Scandura's [27] structural learning. The use of computational models derived from artificial intelligence studies is particularly promising [1, 3, 15], as are those models based on psychological principles with systemic foundations such as personal construct psychology [28, 29, 8, 11, 14]. The explanation of problems of learning structure from behavior in terms of complexity-theoretic measures [6, 7] suggests that their application to VM-problem relations may be valid as a measure of psychological difficulty also.

The User in the Virtual Machine Hierarchy

The VM hierarchy may also be extended to include the user. Nelson [23] has emphasized that the user sees the virtual machine not the underlying one, and Smith [30] has noted this as a design principle of the Xerox Star:

"if everything in a computer system is visible on a display screen, the display becomes reality."

We tend to treat the user as the end of the VM hierarchy and concentrate on the user interface to the machine. However, a more general viewpoint is to treat the machine plus its user as a new virtual machine, just as we treated the machine plus program as a new VM. We can then see the user+machine VM as an **extension** of either and realize that the key problem is their joint solution of a common task. The human-computer interface within the new VM has a richer significance in relation to the performance of the common task than it does in isolation.

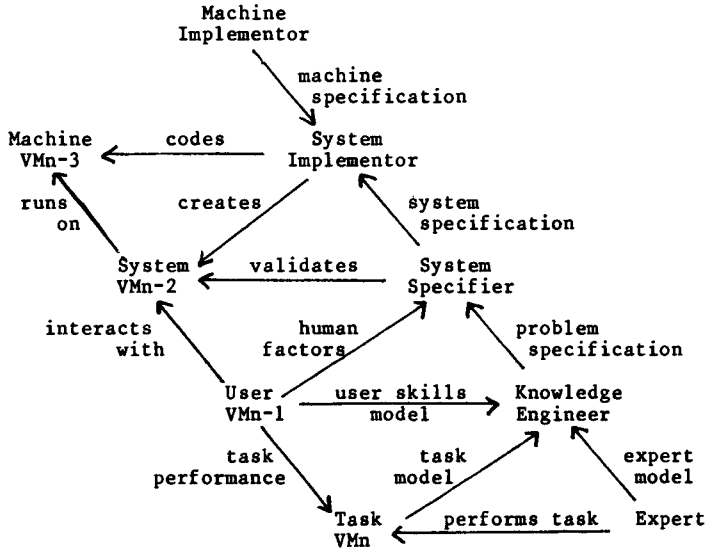

Fig. 4 User and task in the virtual machine hierarchy

Fig. 4 extends the VM analysis further to include the user and the task also as part of the VM hierarchy, perhaps an assistant creating a VM for an executive who wishes to create legal documents. The notion of a "programmer" then extends naturally to the hierarchy of expert, knowledge engineer, system specifier, system implementor and machine implementor shown on the right. The roles and sources of information of all those concerned are clearly differentiated:

the **task** is shown as a VM because in general it may not be an end-result but just another VM, or service provided;

the **expert** is someone who can perform the task without any system support;

the **user** is someone who should be able to perform the same task with system support - note that the "user" is actually VMn-1 consisting of the user interacting interacting with VMn-2 - the model allows for the user to perform the task supported by the computer or the computer to perform it supported by the user, or both;

the **knowledge engineer** draws on an **expert model** of how the expert operates, a **task model** of the task, and a **user skills model** of the expertise already possessed by the user, to define a **problem specification;**

the **system specifier** uses the **problem specification** together with a (problem-independent) **human factors** model of the user to define a **system specification** which he can validate;

the **system implementor** draws on the **system specification** and **machine specification** to create VMn-2 by generating code for VMn-3 on which it runs.

The significance of documentation can be seen in the flows of information on the right hand side. The separation between task-independent human factors and task-dependent user skills models is also made clear. The role of an expert as adjunct to, or alternative to, direct task analysis may also be seen, and is independent of any particular notion of "expert system"; we have always used expert task performers as sources of system design knowledge. The complexity of human factors considerations in overall system design is apparent from the number of differing roles exhibited by people in this figure. This is further exacerbated when it is noted that the figure is fully recursive in that any person on the right hand side may be provided with computer-based support tools and hence become the user on the left hand side of another such figure.

Conclusions

HCI studies are in a transition to deeper theoretical models from which design principles can be derived. The extension of the virtual machine hierarchy to include the people within a system in their various roles provides a framework for analysing all aspects of HCI. In particular the user interacting with a computer system may be seen as forming a new virtual machine. It is suggested that the formal analysis possible for the computational aspects of the VM hierarchy be extended to the human aspects also, and that category-theoretic and complexity-theoretic analyses of cognitive models may produce the theoretical foundations required.

Acknowledgements

Financial assistance for this work has been made available by the National Sciences and Engineering Research Council of Canada.

References

[1] D.G.Bobrow and A.Collins (eds.), **Representation and Understanding: Studies in Cognitive Science.** New York: Academic Press, 1975.

[2] E.A.Edmonds and S.P.Guest, "The SYNICS2 user interface manager," **Interact'84: First IFIP Conference on Human-Computer Interaction,** vol. 1, pp. 53-56, September 1984. Amsterdam: Elsevier.

[3] N.Findler (ed.), **Associative Networks: Representation of Knowledge by Computers.** New York: Academic Press, 1979.

[4] B.R.Gaines, "Varieties of computer - their applications and inter-relationships," **Proceedings of IFAC Symposium on "Pulse Rate and Pulse Number Signals in Automatic Control"**, pp. 1-16, 1968. Budapest.

[5] B.R.Gaines, "Analogy categories, virtual machines and structured programming," in G.Goos and J.Hartmanis (eds.), **GI - 5.Jahrestagung. Lecture Notes in Computer Science,** vol. 34, pp. 691-699, 1975. Berlin: Springer Verlag.

[6] B.R.Gaines, "On the complexity of causal models," **IEEE Transactions on Systems, Man & Cybernetics,** vol. SMC-6, no. 1, pp. 56-59, January 1976.

[7] B.R.Gaines, "Behaviour/structure transformations under uncertainty," **International Journal of Man-Machine Studies,** vol. 8, no. 3, pp. 337-365, May 1976.

[8] B.R.Gaines, "Hardware engineering and software engineering," **Euromicro,** vol. 3, no. 2, pp. 16-21, April 1977.

[9] B.R.Gaines, "The role of the behavioural sciences in programming," in **Structured Software Development Vol.2.** Maidenhead, UK: Infotech International, 1979, pp. 57-68.

[10] B.R.Gaines and P.V.Facey, "Some experience in interactive system development and application," **Proceedings Institution of Electrical and Electronics Engineers,** vol. 63, pp. 894-911, June 1975.

[11] B.R.Gaines and M.L.G.Shaw, "A programme for the development of a systems methodology of knowledge and action," in W.J.Reckmeyer (ed.), **General Systems Research and Design: Precursors and Futures.** Louisville, Kentucky: Society for General Systems Research, January 1981, pp. 255-264.

[12] B.R.Gaines and M.L.G.Shaw, "Analysing analogy," in R.Trappl, L.Ricciardi and G.Pask (eds.), **Progress in Cybernetics and Systems Research Vol.IX.** Washington: Hemisphere, 1982, pp. 379-386.

[13] B.R.Gaines and M.L.G.Shaw, "Dialog shell design," **Proceedings of INTERACT'84: 1st IFIP Conference on Human-Computer Interaction,** vol. 1, pp. 344-349, September 1984. Amsterdam: Elsevier.

[14] B.R.Gaines and M.L.G.Shaw, "Foundations of expert systems," in J.Kacprzyk and R.R.Yager (eds.), **Management Decision Support Systems Using Fuzzy Sets and Possibility Theory.** Rheinland: ISR Interdisciplinary Systems Reserch, Verlag TUV, 1984.

[15] D.Gentner and A.L.Stevens (eds.), **Mental Models.** Hillsdale, New Jersey: Lawrence Erlbaum, 1983.

[16] P.Gordon, "What is integrated software?," **PC World,** vol. 2, no. 11, pp. 72-77, October 1984.

[17] W.J.Hansen, "User engineering principles for interactive systems," **Proceedings of the Fall Joint Computer Conference,** vol. 39, pp. 523-532, 1971. New Jersey: AFIPS Press.

[18] L.R.Harris, "Experience with INTELLECT," **AI Magazine,** vol. 5, no. 2, pp. 43-50, 1984.

[19] R.J.K.Jacob, "Executable specifications for a human-computer ," in A.Janda (ed.), **Human Factors in Computing Systems: CHI'83 Conference Proceedings,** pp. 28-34, December 1983. Boston.

[20] A.Kidd, **Man-Machine Dialogue Design.** BTRL, Ipswich, UK: Martlesham Consultancy Services, 1982.

[21] J.Martin, **Design of Man-Computer Dialogues.** New Jersey, USA: Prentice-Hall, 1973.

[22] T.P.Moran, "The command language grammar: a representation for the user interface of interactive computer systems," **International Journal of Man-Machine Studies,** vol. 15, no. 1, pp. 3-50, July 1981.

[23] T.Nelson, "Interactive systems and the design of virtuality," **Creative Computing,** vol. 6, no. 11, pp. 56-62, November 1980.

[24] R.S.Nickerson, "Man-computer interaction: a challenge for human factors research," **IEEE Transactions on Man-Machine Systems,** vol. MMS-10, no. 4, pp. 164-180, December 1969.

[25] R.S.Nickerson, "Why interactive computer systems are sometimes not used by people who might benefit by them," **International Journal of Man-Machine Studies,** vol. 15, no. 4, pp. 469-483, November 1981.

[26] D.A.Norman, "Design principles for human-computer interfaces," in A.Janda (ed.), **Human Factors in Computing Systems: CHI'83 Conference Proceedings,** pp. 28-34, December 1983. Boston.

[27] J.M.Scandura, **Structural Learning. I. Theory and Research.** New York: Gordon & Breach, 1973.

[28] M.L.G.Shaw, **On Becoming a Personal Scientist: Interactive Computer Elicitation of Personal Models of the World.** London: Academic Press, 1980.

[29] M.L.G.Shaw (ed.), **Recent Advances in Personal Construct Technology.** London: Academic Press, 1981.

[30] D.C.Smith, "Visibility in the Star user interface," **Proceedings of Graphics Interface '83,** p. 181, 1983. Ottawa: National Research Council of Canada.

[31] D.C.Smith, C.Irby, R.Kimball, B.Verplank and E.Harslem, "Designing the Star user interface," in P.Degano and E.Sandewall (eds.), **Integrated Interactive Computing Systems.** Amsterdam: North-Holland, 1983, pp. 297-313.

[32] B.H.Thompson, F.B.Thompson and T.-P.Ho, "Knowledgeable contexts for user interaction," **Proceedings of National Computer Conference,** vol. 52, pp. 29-38, 1983. Arlington, Virginia: AFIPS Press.

[33] G.H.Walther and H.F.O'Neil, "On-line user-computer interface - the effects of interface flexibility, terminal type, and experience on performance," **Proceedings of the National Computer Conference,** vol. 43, pp. 379-384, 1974. New Jersey: AFIPS Press.

[34] H.J.Weegenaar, "Virtuality and other things like that," **Proceedings of IEEE COMCON,** vol. CH1388-8/78, pp. 287-293, 1978.

[35] W.B.Weimer and D.S.Palermo (eds.), **Cognition and the Symbolic Process.** Hillsdale, New Jersey: Lawrence Erlbaum, 1974.

[36] B.H.Williges and R.C.Williges, "Dialogue design considerations for interactive computer systems," in F.A.Muckler (ed.), **Human Factors Review: 1984.** Santa Monica, California: Human Factors Society, 1984, pp. 167-208.

[37] G.H.Woodmansee, "The VisiOn experience - from concept to marketplace," **Proceedings of INTERACT'84: 1st IFIP Conference on Human-Computer Interaction,** vol. 2, pp. 162-166, September 1984. Amsterdam: Elsevier.