Developments in dialog engineering

by MILDRED L. G. SHAW and BRIAN R. GAINES
York University
Toronto, Canada

ABSTRACT

The human-computer interface (HCI) is increasingly the major determinant of the success or failure of computer systems. It is time that foundations of dialog engineering for HCI were provided as explicit and well-founded as those for hardware and software engineering. HCI entered its period of theoretical consolidation at the beginning of the fifth generation in 1980. The lists of pragmatic dialog rules for HCI in the fourth generation have served their purpose, and effort should now be directed to the underlying foundations. Through the influences of other disciplines and their contribution to software engineering, a rich environment for HCI studies, theory, and applications now exists. This can be explored systematically by analyzing the various analogies to HCI possible when the parties are taken to be general systems, people, or computers. The fundamental principles at different levels may be used in the practical design of dialog shells for engineering effective dialog.

INTRODUCTION

This paper is concerned with techniques and underlying principles for *dialog engineering*;[1] that is, for programming effective human-computer interaction (HCI). Interest in HCI has existed since the early days of computing, became highly significant in applications as interactive computing came into widespread use in the 1960s, and is fundamental to fifth-generation computing system (FGCS) development. Users and commentators have been ambivalent towards the state of the art in HCI, oscillating between the euphoric prospects of human-computer symbiosis, and the frustratingly poor human-computer interfaces provided with many systems. An overview of the issues involved can be gained through a brief historic summary.

Computers are tools to enhance the capabilities of people. Human factor considerations have always played a role in computing system design. Babbage made extensive studies of logarithm table legibility with various combinations of type size, and ink and paper colors. Mauchly, in his 1947 discussion on the role of subroutine facilities in EDVAC programming, remarked that any machine coding system should be judged on how easy it is for the operator to obtain results.[2]

In the early years of computing, the machines were so slow, expensive, and unreliable that interactive use was rare in all but a few military and control applications. Those interacting with the machines were the few skilled operators who accepted the problems of interaction as minor compared to all the other difficulties of using computers. Professional ergonomic considerations of computer consoles date from Shackel's paper of 1959,[3] but it was the advent of timesharing systems in the early 1960s that made interactive access widely available and focused attention on human factors. In 1963/64 the MIT MAC,[4] RAND JOSS,[5] and Dartmouth College BASIC[6] systems pioneered a new style of computing whereby the problems of HCI became significant.

Timesharing systems opened computer use to a wider community that was less tolerant of technical problems. As early as 1967, Mills foresaw a future in which professionals would be among the least numerous and least significant system users.[7] White, in considering the problems of user communications said, "The user of many systems soon gets the feeling that either the messages he receives were hung onto the sytem as an afterthought or that they were designed for the convenience of the system rather than the user."[8] Unfortunately, this remark has not gone stale with age and is still echoed in the reviews of software packages.

In 1969, Nickerson surveyed work on man-machine interaction, remarked on its paucity, and noted the lack of contribution of psychology as a discipline to the design of interactive systems.[9] The problems created by poor HCI were highlighted by Walther and O'Neill at the NCC in 1974 where they noted the frustration of casual users with the unyielding, rigid, and intolerant dialog of time-sharing services, and its resultant damage to the industry.[10] Nickerson's 1981 paper on the same theme shows that many problems remain to be solved.[11]

The culmination of this emphasis on HCI may be seen in the Japanese announcement in 1981 of a program of development for a *fifth generation* of computing systems, and the funding of the ICOT research center in Tokyo.[12] Karatsu emphasized the HCI theme of this work by stating, "Until today, man had to walk toward the machine with effort to fit precisely. On the contrary, tomorrow, the machine itself will come to follow the order issued by man."[13]

At the start of the fifth generation forty years after the birth of computers, we know much about the human-computer interface, and we have many examples of good HCI, but we still produce systems that are extremely poor. Fortunately, the market place is coming to dominate the evolution of software and ensure the survival of only that with good HCI. Decreasing costs and the technologies of tomorrow will allow the complexity of the interface to increase, and the richness of multi-modal interaction through multi-media interfaces will place new demands on our HCI design capabilities.

It is time to provide foundations of dialog engineering for the human-computer interface as explicit and well-founded as those for hardware and software engineering. We have case histories and pragmatic rules, but not the theoretical framework within which the case histories can be understood and the rules developed. This paper outlines the basis for developing this framework during the next few years. The next section discusses how styles of dialog have been evolving. The following section systematically examines various sources of theoretical foundations. The final section discusses how these come together in the design of dialog shells.

STYLES OF INTERACTION

Hansen's comparison of the interactive system designer with a composer of music[14] appears very apt when one examines the wide variety of styles that have been developed for HCI. People and computers are very different systems; there are no universally obvious ways in which they should interact. Different designers have different views, often strongly held, and different users have different preferences, often equally strong. Three main styles of dialog may be distinguished.[15]

1. *Formal dialog,* in which the activities and data structures within the computer are presented externally in a direct

representation with the minimum syntactic sugar necessary to aid human cognition;

2. *Natural language dialog,* in which the human use of language to communicate information and commands is simulated within the narrow context of the activities and data structures within the computer;
3. *Graphic dialog,* in which the human manipulation of objects to communicate information and commands outside of the computer is simulated to access the activities and data structures within the computer.

The *prompt-response* dialogs of interactive JCLs such as those of VMS, Unix and CP/M are typical of formal dialogs. Zloof's *query-by-example* database retrieval system is a prime example of the simplicity and power of well-designed formal dialog.[16] It presents the structure of a relational database directly to the user with a minimum of syntactic sugar. The user can see what is there, the system is reasonably clear to naive users, and it is eminently natural to professional users.

There has always been ambivalence towards *natural language* dialog because it is remote from the actual operation of current computer systems.[17] Natural language is attractive because computer users already have the skills to use it to communicate with other people. However, we may risk that the very partial simulations of people that we can program currently are taken by users to have the full capabilities of people because they can "understand" natural language. In sociological terms, we may not be able to maintain the *role integrity** of a computer system with a natural language interface. In practice, commercial experience with such packages as Harris' *INTELLECT* indicate that they are attractive to users, and that problems of role integrity are not significant in applications to date.[18] It is probably more relevant to contrast applications where a concise formal dialog is adequate and more efficient than natural language with those where requirements are very open-ended and the complex formal language that might be required is best generated by translation from natural language. This is now feasible even on microcomputers using natural language shells such as *ASK.*[19]

Graphic dialog might more generally be termed *object simulation* because the graphic capabilities of the computer are used to create a model world that mimics the characteristics of the real world to the user. The screen of a Xerox *Star* simulates a desk on which are folders containing documents, in- and out-trays, and a trash bin.[20] Similar considerations apply to graphic dialog as to natural language dialog. Graphic dialog is attractive because it also brings into play the existing skills of a user, for example in dealing with an office environment and in the *direct manipulation*[21] of objects, but it becomes very important to ensure that all the user's expectations in that environment are properly fulfilled.

Within each of these styles of interaction, there has been a developmental sequence such that we can distinguish levels of capability. Figure 1 tabulates the development of styles of dialog from generations two through five of computing systems.

*Role integrity is consistency in maintaining the behavioral patterns associated by others with the role they perceive.

	GRAPHIC DIALOG Representing world	FORMAL DIALOG Representing computer	NATURAL LANGUAGE DIALOG Representing person
2 1956 -63	EXPENSIVE Flight simulators & process simulators for limited military & industrial use	SIMPLISTIC Job control languages giving operator access to machine features	TRIVIAL Natural language output from stored script in early computer-assisted instruction.
3 1964 -71	PRACTICAL Mimic diagrams. Light pens. Touch screens.	STANDARD Simple prompt-response. Menus. Form-filling.	PRIMITIVE Keyword recogntion. Incorporation of words input into output.
4 1972 -79	CREATING REALITY Windows. Icons. Desktop simulation.	SOPHISTICATED Dialog-engineered P-R. Interactive F-F. Intelligent F-F.	PRACTICAL Understanding fixed domain. Understanding DBMS domain. Meta-level understanding.
5 1980 -87	LOW-COST PC flight simulators.	INTEGRATIVE < * * * Integration through ASK and INTELLECT * * * > < * * * * * Integration through Macintosh * * * * * > < * * Integration through Symphony & Framework * * > < * * * * * * * * * * * * * LISP machine applications * * * * * * * * * * * * * > < * * * * * * * * * * * * Fifth generation objectives * * * * * * * * * * * * >	SOPHISTICATED

Figure 1—Development of styles of interactive dialog

The center column shows formal dialog representing the computer characteristics directly developing from the early job control languages through simple prompt-response sequences suited to teleprinters; menus as visual displays became available; form-filling as cursor-addressable displays became available for use in transaction processing; dialog-engineered prompt-response with *HELP* facilities, default entries, and so on, as user support began to be considered; interactive form-filling as transaction processing became interactive; and intelligent form-filling with fields filled in through calculations as Visicalc was invented at the end of this era.

The left column shows graphic dialog developing from expensive applications in limited military and industrial systems through mimic diagrams, light pens and touch screens; the windows, icons and artificial reality of Smith's *Pygmalion*[22] which led to the Xerox *Star;* the decline in cost of computer hardware allowing flight simulation on personal computers; and a variety of integrated systems.

The right column shows natural language dialog developing from the trivial output of stored text in early computer-assisted instruction (CAI) systems; keyword recognition in later CAI; incorporation of input text in output as in Weizenbaum's *ELIZA*[23] and its practical application in Shaw's *PEGASUS;*[24] understanding a fixed domain as in Winograd's *SHRDLU;*[25] understanding a dynamic database as in Harris' *ROBOT*[26] (later *INTELLECT*); understanding the process of understanding itself as in Davis' TEIRESIAS;[27] and a variety of integrated systems.

We are now in the fifth generation era where the divisions between styles of dialog are beginning to break down as systems become increasingly integrated. The Apple *Macintosh* integrates graphic dialog in a simulated desk-top environment with the formal dialog of pull-down menus. Packages such as Lotus *Symphony* and Ashton-Tate's *Framework* integrate graphic and formal dialog in a variety of combinations. The natural language shells already mentioned, *ASK* and *INTELLECT,* translate natural user queries into formal DBMS enquiries. The advanced architecture of *LISP machines* supports high-resolution graphics, windows and icons, and is a natural

tool for work on all aspects of HCI including natural language. Such machines are the first examples of systems that begin to satisfy the FGCS objectives.

SYSTEMATIC ANALYSIS OF PRINCIPLES FOR EFFECTIVE DIALOG

As experience in the design and use of interactive systems grew, various authors presented guidelines for the design of effective human-computer dialog. Hansen at the 1971 FJCC seems to have been the first to develop user engineering principles for interactive systems.[14] The number of guidelines for dialog engineering has grown over the years. Ours started in 1975 as a set of 11 rules for programming interactive dialog,[28] growing to 30 in 1984.[15] They were first formulated for interaction through low-speed, upper-case teleprinters, but generalized well to visual displays and dialog through menus and forms. They appear to generalize adequately to dialog through windows, icons, and mice, and through restricted natural language. However, the proliferation of rules (a 1984 paper has over 500[29]), the development of new technologies for HCI, the growing maturity of studies of human cognition, human-computer interaction, and computer communication protocols, the need for standard dialog engineering tools, and the emergence of commercial products to satisfy that need suggests that the time is ripe for a new approach to dialog engineering.

All technologies go through phases after the initial breakthrough in which experience is gained, design rules are derived from experience, and theoretical foundations are derived to predict experience and derive design rules.[30] The breakthrough for HCI was the development of timesharing at the beginning of the third generation in 1964. Pragmatic rules were developed during the fourth generation from 1972 to 1979. During the fifth generation from 1980 to 1987, we should be seeing theoretical foundations develop for HCI. This theory must have its foundations in the larger context of studies of applied psychology, linguistics, computer communication, and general systemic principles. Although it may require fresh validation in the context of HCI, it has richer origins and support than studies of HCI alone. The following sections explore the wider context systematically.

System-system Interaction

One useful technique in analyzing HCI is generalization, noting that people and computers are both examples of *systems*. Systems can interact in many ways, but there are constraints on the type of interaction that we consider in HCI. In particular, we would expect that at least one of the systems will be *goal-seeking* and that the satisfaction of its goals, and the cost in doing so, will give us a basis for *evaluating* the interaction. The goal of one system may be to transfer information from the other through *communication*, to predict the behavior of the other through *modeling*, or to change the state of the other through *control*. Both systems may be goal-seeking, in which case considerations of *co-operation* and *competition* arise. The goal of one system may be to aid the other in, or to prevent it from communication, modeling or control.

There are many system-theoretic principles which are instantiated in the situation where one system is a person and the other a computer. Wiener emphasized this in his development of *cybernetics* as the study of "communication and control in men and machines."[31] Many principles of communication theory and control theory apply directly to HCI; for example, the dialog rule that *the user should dominate the computer*[1] is derived from a stability-theoretic result in control theory, that two coupled systems with similar time constants may oscillate unstably around their intended equilibrium state (i.e., the person modeling a computer and adapting to it while the computer is modeling the person and adapting to him is a potential source of mutual instability).

The system-theoretic foundations for dialog engineering are particularly important as technologies change. Concrete rules are needed that can be applied within a range of contexts without excessive development and inference, but when the context varies, the rule may become invalid. Indeed, the negation of a rule may become valid instead. The system-theoretic foundation for the rule is then essential in order to enable the variant appropriate to the new context to be derived. There are situations where it is appropriate for the computer to dominate the interaction, notably ones where the person is not able to adapt, but where the computer is able to do so to improve the interaction. An example might be the indexing of very large databases that a given person accesses infrequently and does not learn to navigate. The computer might then adapt its portrayal of the indexing material to the nature of the inquiry without ill-effects.[32]

Computer-system Interaction

If one of the interacting systems is taken to be a computer, then HCI can be seen to be analogous to interfacing a computer to another system such as a piece of equipment. The design principles applicable to computer-equipment interfaces are well known and carry over to person-computer dialog. Problems arise because the system to which the computer is to be interfaced already exists and is not another programmable computer. We may have to take it as it is and design an interface that copes with its peculiarities. The dialog rule *use the user's model*[1] derives from the idea that the dialog engineer should identify the existing interface and attempt to emulate it rather than change it. Problems also commonly arise through noise at the interface. The designer attempts both to provide a low-noise channel and to provide error-detection and correction for unavoidable noise. In HCI, such noise may arise through lack of clarity in information presentation giving rise to perceptual errors in one direction, mis-keying giving rise to errors in the other direction, and so on. The dialog rule *validate data on entry*[1] is a principle of communication over a noisy channel.

Computer-computer Interaction

When the interacting systems are both computers, it is possible to define *protocols* that may be implemented by any

programmable digital system, and dialog engineering rules that may be seen as defining a *human protocol.*[33] The *Open System Interconnection* (OSI) ISO standard[34] is particularly interesting because it hierarchically structures computer-computer protocols for networks in a way that may have relevance for person-computer protocols. The concept of an *open system* is in itself relevant because it expresses objectives for computer networks that are equally applicable to people using those networks. The aim is to allow integrated systems to be formed from multiple components not all from one vendor and not all installed at the same time.[35] The OSI concept is that the network is open to all systems that conform in their communications with certain well-defined protocols. In human terms, the protocols may be seen as social norms for the behavior of members of a club; anyone may join provided they agree to conform to these norms.

The OSI standard is hierarchical, defining a number of layers, each with its own standard. Each of these levels has an analogy in human information processing and communication, e.g., the physical layer corresponds to the audio-visual perceptual processes, and there is much to be gained in applying the OSI concepts to the human protocol.[36]

Person-system Interaction

When one of the systems is a person we have the classic case of man-machine interaction. Considerations of people interacting with equipment has been treated as a branch of applied psychology termed *ergonomics* that arose under the same pressures as computer technology out of World War II studies of pilots, gunners, and so on. There is a wealth of results on general problems of human skills, training, its transfer between different learning situations, the effects of fatigue, and so on, that is immediately applicable to HCI. While interactive computers have been used primarily as programming and data entry systems, these effects have not been major considerations. However, as computer-based interfaces become increasingly the norm for a wide variety of human activities, the classic results of applied psychology and ergonomics are becoming increasingly important. The novelty of the computer should not blind us to its commonality with much earlier equipment. We do not have the time and effort to waste on rediscovering what is already known.

Person-person Interaction

When both systems are people, we have normal linguistic interaction from which the terms human-computer "conversation" and "dialog" have been generalized. Modern linguistic theory has become increasingly concerned with the interaction between participants in a dialog, rather than a view of linguistic output as a predefined stream to be decoded.[37] This provides a rich source of models for person-computer interaction, particularly as AI techniques take the computer closer to emulating people and their language behavior. There are also useful analogies to be drawn between the behavior of casual computer users and the transactional analysis of the behavior of strangers meeting.[38] Many of the principles of language such as *scoping* and *pronomial reference* are general principles of communication that occur in the graphic/gesture dialog with computers involving no textual interaction.[36]

DIALOG SHELLS

The considerations of the previous section suggest a hierarchy of levels of dialog rule generation (see Figure 2) including systemic principles, particular features of people and computers, considerations of their interaction, styles or technologies of interaction which lead to dialog shells, and particular applications.

The interaction level is interesting because it is where systemic person-and-computer principles come together. For example, the dialog rule *avoid acausality*[1] has a system-theoretic foundation in that causal modeling systems generate meaningless models of systems with even slight acausalities. However, to apply this principle, we have to know that people are causal modelers,[39] and we should not regard the rule as significant unless we know that time-sharing systems generate apparently random delay distributions.

At the instantiation level, the rules are applied to instances of styles and technologies of interaction. To make them application-independent, there has been a move to incorporate the rules in dialog shells that interface between the user and the computer system. The first general shells such as IBM EXEC originated from job control languages. A shell may incorporate many of the formal dialog rules for prompt-response.[15] Shells for form-filling transaction processing and menu-based systems are in widespread use. Several commercial companies have introduced window shells for the IBM PC, and Apple Macintosh has a general icon/mouse shell. *INTELLECT* is a natural language shell for mainframes, and *ASK* is one for micros. Expert system shells such as *EMYCIN* and *AL/X* are recent developments that incorporate signifi-

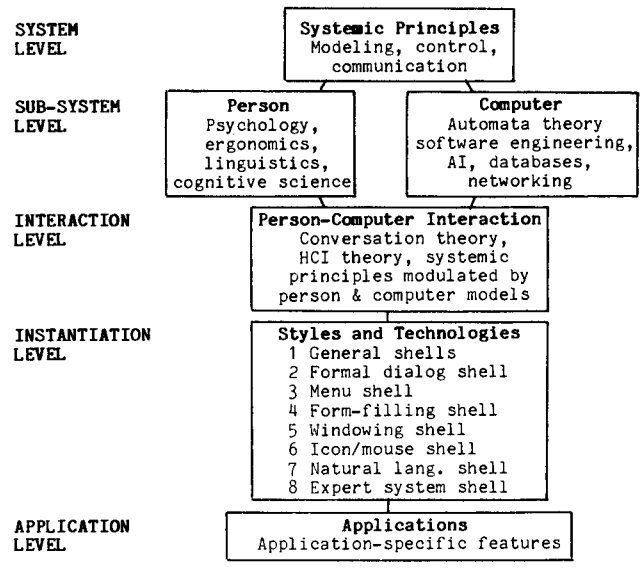

Figure 2—Levels of dialog rule generation

cant new dialog techniques such as the uniform availability of *why* and *how* queries.[40]

CONCLUSIONS

The human-computer interface is increasingly the major determinant of the success or failure of computer systems. Its improvement is a major objective of the fifth generation computing development program. The literature from early days of computing to the present shows that the problems of HCI have been recognized but still adversely affect the use of computers. It is time that we provided foundations of dialog engineering for the human-computer interface as explicit and well-founded as those for hardware and software engineering. These may be based on the systemic, computational, psychological, and linguistic principles underlying HCI. They will then lead to the design of a variety of general-purpose dialog shells.

ACKNOWLEDGMENTS

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

REFERENCES

1. Gaines, B. R. and M. L. G. Shaw. "Dialog Enegineering." in Sime, M., and M. J. Coombs (eds.), *Designing for Human-Computer Interaction.* London: Academic Press, 1983, pp. 23–53.
2. Mauchly, J. W. "Preparation of Problems for EDVAC-type Machines." in Randell, B. (ed.), *The Origins of Digital Computers: Selected Papers.* Berlin: Springer-Verlag, 1973.
3. Shackel, B. "Ergonomics for a Computer." *Design,* 120 (1959), pp. 36–39.
4. Fano, R. M. "The MAC System: A Progress Report." in Sass, M. A. and W. D. Wilkinson (eds.), *Computer Augmentation of Human Reasoning.* Washington: Spartan Books, 1965, pp. 131–150.
5. Shaw, J. C. "JOSS: Experience with an Experimental Computing Service for Users at Remote Consoles." in Orr, W. D. (ed.), *Conversational Computers.* New York: John Wiley, 1968, pp. 15–22.
6. Danver, J. H. and J. M. Nevison. "Secondary School Use of the Time-shared Computer at Dartmouth College." *AFIPS, Proceedings of Spring Joint Computer Conference* (Vol. 34), 1969, pp. 681–689.
7. Mills, R. G. "Man-machine Communication and Problem Solving." in Cuadra, C. A. (ed.), *Annual Reviews of Information Science and Technology, 2.* New York: Interscience, 1967.
8. White, R. R. "On-line Software—the Problems." in Gruenberger, F. (ed.), *The Transition to On-Line Computing.* Washington: Thompson, 1967, pp. 15–26.
9. Nickerson, R. S. "Man-computer Interaction: A Challenge for Human Factors Research." *IEEE Transactions on Man-Machine Systems,* MMS-10 (1969), 4, pp. 164–180.
10. Walther, G. H. and H. F. O'Neil. "On-line User-computer Interface—the Effects of Interface Flexibility, Terminal Type, and Experience on Performance." *AFIPS Proceedings of the National Computer Conference* (Vol. 43), 1974, pp. 379–384.
11. Nickerson, R. S. "Why Interactive Computer Systems are Sometimes Not Used by People Who Might Benefit by Them." *International Journal of Man-Machine Studies,* 15 (1981), 4, pp. 469–483.
12. Moto-Oka, T. (ed.), *Fifth Generation Computer Systems.* Amsterdam: North-Holland, 1982.
13. Karatsu, H. "What is Required of the Fifth Generation Computer—Social Needs and Impact." in Moto-oka, T. (ed.), *Fifth Generation Computer Systems.* Amsterdam: North-Holland, 1982, pp. 93–106.
14. Hansen, W. J. "User Engineering Principles for Interactive Systems." *AFIPS Proceedings of the Fall Joint Computer Conference* (Vol. 39), 1971, pp. 523–532.
15. Gaines, B. R. and M. L. G. Shaw. *The Art of Computer Conversation: A New Medium for Communication.* Englewood Cliffs, N.J.: Prentice-Hall, 1984.
16. Zloof, M. M. "Office-by-example: A Business Language that Unifies Data and Word Processing and Electronic Mail." *IBM Systems Journal,* 21 (1982), 3, pp. 272–304.
17. Hill, I. D. "Natural Language Versus Computer Language." in Sime, M. and M. J.Coombs (eds.), *Designing for Human-Computer Interaction.* London: Academic Press, 1983, pp. 55–72.
18. Harris, L. R. "Experience with INTELLECT." *AI Magazine,* 5 (1984), 2, pp. 43–50.
19. Thompson, B. H., F. B. Thompson, and T.-P. Ho. "Knowledgeable Contexts for User Interaction." *AFIPS, Proceedings of the National Computer Conference* (Vol. 52), 1983, pp. 29–38.
20. Smith, D. C., C. Irby, R. Kimball, B. Verplank, and E. Harslem. "Designing the Star User Interface." In Degano, P., and E. Sandewall (eds.), *Integrated Interactive Computing Systems.* Amsterdam: North-Holland, 1983.
21. Shneiderman, B. "Direct Manipulation: A Step Beyond Programming Languages." *Computer,* 16 (1983), 8, pp. 57–69.
22. Smith, D. C. *Pygmalion.* Basel, Switzerland: Birkhauser, 1977.
23. Weizenbaum, J. *Computer Power and Human Reason: From Judgement to Calculation.* San Francisco: W. H. Freeman, 1976.
24. Shaw, M. L. G. *On Becoming a Personal Scientist.* London: Academic Press, 1980.
25. Winograd, T. *Undersanding Natural Language.* Edinburgh: Edinburgh University Press, 1972.
26. Harris, L. R. "User Oriented Data Base Query with the ROBOT Natural Language Query System." *IJMMS,* 9 (1977), 6, pp. 697–713.
27. Davis, R., and D. B. Lenat. *Knowledge-Based Systems in Artificial Intelligence.* New York: McGraw-Hill, 1982.
28. Gaines, B. R., and P. V. Facey. "Some Experience in Interactive System Development and Application." *IEEE Proceedings,* 63 (1975), pp. 155–169.
29. Williges, B. H., and R. C. Williges. "Dialogue Design Considerations for Interactive Computer Systems." in Muckler, F. A. (ed.), *Human Factors Review: 1984.* Santa Monica: Human Factors Society, 1984, pp. 167–208.
30. Gaines, B. R., and M. L. G. Shaw. "Generations of Computers: Modeling and Forecasting." *Possible Worlds.* 1–4 (1985), pp. 3–16.
31. Wiener, N. *Cybernetics.* Cambridge, Mass.: MIT Press, 1948.
32. Witten, I. H., S. Greenberg, and J. Cleary. "Personalizable Directories: A Case Study in Automatic User Modelling." *Proceedings of Graphics Interface '83.* Ottawa: National Research Council of Canada (1983), pp. 183–189.
33. Shaw, M. L. G., and B. R. Gaines. "Does the Human Component in a Network Have a Protocol?" *Proceedings of the International Electrical, Electronics Conference and Exposition.* IEEE 83CH1955-4 (1983), pp. 546–549.
34. Day, J. D., and H. Zimmerman. "The OSI Reference Model." *IEEE Proceedings,* 71 (1983), 12, pp. 1334–1340.
35. Gaines, B. R. "From Word Processing to Image Processing in Office Systems." *Proceedings of the International Electrical, Electronics Conference and Exposition.* IEEE 83CH1955-4 (1983), pp. 622–625.
36. Taylor, M. M., C. A. McCann, and M. I. Tuori. "The Interactive Spatial Information System." *DCIEM Report 84-R-22.* Toronto: DCIEM, 1984.
37. Bennett, J. *Linguistic Behaviour.* Cambridge, UK: Cambridge University Press, 1976.
38. Berne, E. *What Do You Do After You Say Hello?* London: Andre Deutsch, 1974.
39. Gaines, B. R. "On the Complexity of Causal Models." *IEEE Transactions on Systems, Man & Cybernetics.* SMC-6 (1976), 1, pp. 56–59.
40. Gevarter, W. B. "Expert Systems: Limited but Powerful." *IEEE Spectrum,* 18 (1983), pp. 39–45.