Int. J. Man-Machine Studies (1986) **24**, 101-123

Foundations of dialog engineering: the development of human–computer interaction. Part II

BRIAN R. GAINES

Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4

AND

MILDRED L. G. SHAW

Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3

(Received 8 February 1985 and in revised form 10 October 1985)

The human-computer interface is increasingly the major determinant of the success or failure of computer systems. It is time that we provided foundations of engineering human-computer interaction (HCI) as explicit and well-founded as those for hardware and software engineering. Through the influences of other disciplines and their contribution to software engineering, a rich environment for HCI studies, theory and applications now exists. Many principles underlying HCI have systemic foundations independent of the nature of the systems taking part and these may be analysed control-theoretically and information-theoretically. The fundamental principles at different levels may be used in the practical design of dialog shells for engineering effective HCI. This paper surveys the development of styles of dialog through generations of computers, the principles involved, and the move towards integrated systems. It then systematically explores the foundations of HCI by analysing the various analogies to HCI possible when the parties are taken to be general systems, equipment, computers or people.

1. Introduction

This paper is the second of two concerned with the development of human-computer interaction. The first paper (Gaines & Shaw, 1986) has suggested that we have been in an era of transition from a period in which HCI concerns were the province of a few specialist researchers to one in which HCI science and technology are widely regarded as basic concerns in computer-based system design and application. It assessed the state of the art in HCI research and development using an analysis of past trends to indicate future directions. It reviewed the development of HCI and related topics in artificial intelligence from the early generations of computers through the present fifth generation and projected future trends. It analysed the place of HCI research, science and technology within the infrastructure of the computing industry, its role in fifth-generation research and commercial product innovation, and the significance of human factors knowledge in the proposed sixth-generation research program.

The first paper noted that one objective of HCI studies is closely related to those of hardware and software engineering and is concerned with the problems of interfacing people to information-processing technology. Another objective of HCI studies is closely related to those of artificial intelligence and is concerned with the problems of

0020-7373/86/020101+23$03.00/0 © 1986 Academic Press Inc. (London) Limited

establishing symbiotic partnerships of people and knowledge-processing technology. Achieving the first is also an essential prerequisite to achieving the second. We need effective human–computer communication before we can achieve effective amplification of intelligence. These two objectives of HCI studies are consistent with the objectives of the sixth-generation computing system development program based on interdisciplinary collaboration between brain physiologists, psychologists, linguists, logicians and computer scientists to develop foundations for knowledge science and technology (STA, 1985). This second paper considers the first objective in more detail, analysing the foundations of dialog engineering from historical and interdisciplinary perspectives, and surveying topics relating to future developments in HCI.

The following section analyses the detailed development of styles of dialog through generations 2–5 of computers, establishing the existing state of the art and projecting future trends. The next section examines the interdisciplinary foundations of HCI studies to project the sources of the theoretical framework seen as vital to fifth- and sixth-generation objectives.

2. Styles of interaction

Hansen's (1971) comparison of the interactive system designer with a composer of music appears very apt when one examines the wide variety of styles that have been developed for HCI. People and computers are very different systems and 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 (Gaines & Shaw, 1984*a*; Shaw & Gaines, 1985*b*) as shown in Fig. 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;

—*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;

—*Graphic dialog* in which the human manipulation of objects within the world to communicate information and commands is simulated to access the activities and data structures within the computer.

FORMAL DIALOG

The *prompt-response* dialogs of interactive JCLs such as those of VMS, Unix and CP/M are typical of formal dialogs. Zloof's (1982) *query-by-example* database retrieval system is a prime example of the simplicity and power of well-designed formal dialog. It presents the structure of a relational database directly to the user with the minimum syntactic sugar. What the user sees is what is there and the system is reasonably clear to naive users and eminently natural to professional users.

NATURAL LANGUAGE DIALOG

There has always been ambivalence towards *natural language* dialog because it is remote from the actual operation of current computer systems (Hill, 1983). Natural

	Graphic Dialog *Representing world*	Formal Dialog *Representing computer*	Natural Language Dialog *Representing person*
2 1956-63	**Expensive** Flight simulators & process simulators for limited military and industrial use	**Simplistic** Job control languages giving operators access to machine features	**Output Only** Natural language output from stored script in early computer-aided instruction
3 1964-71	**Practical** Mimic diagrams Light pens Touch screens	**Standard** Simple prompt-response Menus Form-filling	**Primitive** Keyword recognition Incorporation of words from input in output
4 1972-79	**Creating Reality** Windows Icons Desktop simulation	**Sophisticated** Dialog-engineered prompt-response Interactive form-filling Intelligent form-filling	**Practical** Understanding fixed domain Understanding DBMS domain Meta-level understanding
5 1980-87	**Low-Cost** PC flight simulators Integration through Macintosh Integration through Symphony & Framework LISP machine applications Fifth generation objectives	**Integrative** Integration through INTELLECT and ASK	**Sophisticated**

FIG. 1. The development of styles of dialog during the second to fifth generations of computing.

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's (1984) *INTELLECT* indicate that they are attractive to users, and problems of role integrity are not significant in applications to date. 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.

GRAPHIC DIALOG

Graphic dialog might more generally be termed *object simulation* since the graphic capabilities of the computer are used to create a model world that mimics the characteristics of a real world natural to the user. The classic examples are the Xerox *Star* (Smith, Irby, Kimball, Verplank & Harslem, 1983) and, more recently, the Apple *Macintosh* simulating a desk on which are folders containing documents, in- and

out-trays, and a trash bin. As Smith (1983) notes: "if everything in a computer system is visible on a display screen, *the display becomes reality.*"

Similar considerations apply as for natural language dialog: graphic dialog is attractive because it brings into play the existing skills of a user, e.g. in dealing with an office environment; but it becomes very important to ensure that all the user's expectations in that environment are properly fulfilled.

DEVELOPMENTAL SEQUENCES WITHIN DIALOG STYLES

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 through generations 2–5 of computing systems. 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 (1977) *Pygmalion* which led to the Xerox *Star*; the decline in cost of computer hardware allowing *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 instructed (CAI) systems; *keyword recognition* in later CAI; incorporation of *input text in output* as in Weizenbaum's (1966) *ELIZA* and its practical application in Shaw's (1980) *PEGASUS*; *understanding a fixed domain* as in Winograd's (1972) *SHRDLU*; *understanding a dynamic database* as in Harris's (1977) *ROBOT* (later *INTELLECT*); *understanding the process of understanding* itself as in Davis's *TEIRESIAS* (Davis & Lenat, 1982); and a variety of *integrated systems.*

INTEGRATION OF DIALOG STYLES

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 (Gordon, 1984). The natural language shells, *ASK* (Thompson, Thompson & Ho, 1983) and *INTELLECT* (Harris, 1984), translate natural user queries into formal DBMS enquiries. The advanced architecture of the *Lisp machine* (Bawden, Greenblatt, Holloway, Knight, Moon & Weinreb, 1979; Myers, 1982; Manuel, 1983) supports high-resolution graphics, windows and icons, and is a natural tool for work on all aspects of HCI including natural language. As discussed in the companion paper, the fifth-generation development program involves the development and integration of all three styles of dialog in increasingly natural human–computer interfaces (Moto-oka, 1982).

3. Analysis of dialog styles

The discussion of dialog styles and their development in the previous section is a useful classification and *aide mémoire* for system designers. However, there are some arguments that can be based on it that impact on deeper issues in HCI design and research. These extrapolations are necessarily more speculative than the classification itself. The formal/natural language/graphic dialog classification is based on obvious descriptive and evolutionary aspects of actual dialog systems. In the discussion, however, we introduced a systemic basis for the classification which is worth further exploration.

FORMAL DIALOG, VIRTUAL MACHINES AND SYNTACTIC SUGAR

The main feature of formal dialog is that its structure stems from the underlying virtual machine. It projects the structure of the system programmed and then coats this with the minimum syntactic sugar necessary to make it palatable to a human user. This projection of the actual system is in itself a user aid in helping the user to form a correct model of the system. We avoid the problems of sustaining a, possibly unsustainable, role that may arise if we attempt to simulate a system more natural to the user. However, two major issues then arise: first, given the variety of problem solutions possible that may all be nearly equal in terms of computational space, time, programming effort, and so on, are some solutions more natural to the user than others; second, what is the nature of "syntactic sugar"?

The problem of what, out of many possible programs, is one natural for the user is confounded by the realization that the user will be performing some task. Hence, we have to consider the three-part relation between user, task, and program. In the early days of computing, so-called "analog" computers were attractive in certain simulation problems because their components were analogs of the components in the system being simulated. This made it natural for the user who understood the system being simulated to be able to control the simulation. Such naturality based on relations of analogy seems to apply to spread-sheets and Zloof's query-by-example.

The analysis of syntactic sugar is also interesting since skill in its use is key to effective dialog programming yet we generally treat it very informally. What we do in essence is to take a system whose controls and displays might as well be switches and lamps and make them textual labels instead. The rationale is that the labels are more meaningful and hence easier to understand and remember for the user. The danger is that they have no corresponding meaning for the computer system other than as textual labels. Syntactic sugar is the use of input/output structures that have very much greater semantic content for the user than they do for the computer system. The problem in its use is that there are no natural control structures within the computer system to ensure uniformity and consistency. Arbitrary labels may be used in different ways in different places, or different labels may be used for the same operation. Hence, dialog engineering is required that is distinct from the applications program. In particular, this will include cognitive models of the naturality of command names (Rosenberg, 1983).

NATURAL LANGUAGE DIALOG, ROLE INTEGRITY AND STRICT CONVERSATIONS

The main feature of natural language dialog is that *its structure stems from the linguistic basis of knowledge representation, communication and inference used by people.* It projects

the structure of a person but is unable to sustain the full role of a person. The projected role raises expectations in the user which cannot be fulfilled by the system. If the system is loose in accepting and using input which it cannot fully decode, it may respond on an incorrect basis without the user being aware of it. If the system is tightly constructed to reject input that does not conform to that which it can fully decode then the user may have difficulty in determining which input is acceptable (Ogden & Brooks, 1983).

One approach to overcoming these problems is to decode the input loosely but make it clear to the user how the system has "understood" each communication. *INTELLECT* does this by displaying its translation of each request into the database query language. The assumption is that the formal query language is sufficiently transparent that the user can at least spot any significant errors in translation. This is the equivalent in person-person communication to saying, "This is what I understand you to be saying in my terms and is what I will actually do." *TEIRESIAS* translates a new rule into internal format and then independently translates it back again for the user to check. This is the equivalent in person-person communication to saying, "This is my rephrasing of your statement. Does it still have the meaning you intended?"

Any natural language dialog system at the current state of the art needs to incorporate dialog engineering separately from the application that ensures that the user has feedback on what the system has "understood" from the input. Pask (1980) terms this establishing a *strict conversation* in which one party to a conversation explains to the other its understanding of the information passed between them in such a way as to ensure mutual coherence. There are analogies in lower-level technical systems such as the synchronizing protocols used between high-speed modems. There are also analogies in lower-level conversational acts such as the social protocols of greetings which enable us to identify the phoneme usage of a stranger with an accent differing from our own. It is easy to assume that natural language overcomes the need for dialog engineering because it is "natural". The converse is true. We need to develop protocols for HCI in natural language that support effective communication despite the technical limitations of current natural language processing. This will include the feedback processes of establishing strict conversations and also the definition of coherent domains of naturality between formal and natural language.

GRAPHIC DIALOG, SIMULATION AND REALITY

The main feature of graphic dialog is that *its structure stems from a physical world already familiar to a person.* It projects the structure of a simulated world and attempts to make the dynamics of, and interaction with, this world as close as possible to reality. This creates problems analogous to those of both formal and natural language dialog. We can display the page of a document on the screen of a Star but do not have the space or need to put a waste paper bin there. Instead we use an icon to represent the bin symbolically. The icon is syntactic sugar whose pattern has meaning to the user but acts as only a position marker to the computer. What form of icon is most readily understood correctly by the user? How do we design a set of icons that have an internal coherence such that the user can generalize understanding of some to understanding all? These are dialog engineering questions that are being studied experimentally (Bewley, Roberts, Shroit & Verplank, 1983) and require resolution.

The problem with graphic dialog analogous to that with natural language is that the simulation may deviate from the simulated world because of current technical limita-

tions, errors in the systems analysis or differing user conceptual models. What does the user expect to happen on a Star when he moves a document to a file drawer, and to a printer? In the first case, the original disappears as would a real document. Should it in the second? This is the problem of extending *generic commands* that have a natural physical meaning in many contexts to cases where the meaning is less obvious (Rosenberg & Moran, 1984). The "printer" on the Star operates like the "copier" in an office but the user knows that it is actually a printer. Do we use the behavioral name of the system simulated or the physical name of the system doing the task? Again the gap between simulation and reality requires dialog engineering protocols to be developed for graphic dialog that are able to bridge this gap for the user in a coherent, understandable and transferable way.

THE FUTURE OF DIALOG STYLES

The evolution of dialog styles discussed in this paper has been put in technological terms. Prompt-response dialog is natural for teleprinters. Their slow speed encouraged terse output. Help messages on such slow devices should not be long and hence multi-level help became desirable. Visual displays on high-speed lines allow full page messages to be displayed rapidly and encouraged the development of menus. Cursor-addressing allows random-access to the screen and encouraged form-filling. There are converse arguments, that the development of visual displays was to overcome the HCI problems of teleprinters. However, the evidence is, on balance, that we have been led by the technology. For example, the graphic dialog of the Star became possible through high-resolution graphics. Shortly after this was publicized similar window/icon techniques began to be used in packages on the much older technology of the Apple II. The techniques could have evolved on the earlier system but did not do so. It took the stimulus of new technology to encourage the development of styles that then propagated backwards to the older technology as well as forwards to the new.

If we project future needs for dialog engineering from present trends in technology, then the most obvious gaps are those concerned with multi-modal, multi-channel and multi-user systems. Speech recognition and synthesis and the instrumentation of body posture and gesture give us more modalities for HCI. These, together with multiple windows and multiple screens, give us more channels for parallel, integrated communication. The increasing use of computer-based systems as routine tools in continuous use has already brought into HCI new considerations of vigilance, fatigue, and so on. The use of such systems to tie together groups of users performing related tasks with common goals is bringing into HCI new considerations of small-group structure and social behavior. Dialog engineering for multi-modal, multi-channel, multi-user systems will involve all of the techniques already developed, but also adds new dimensions not yet considered. We have to come to understand the protocols of dialog involving voice and gesture (Taylor, McCann & Tuori, 1984) and the "naturality" of organizational communication paths for different forms of organizational relations.

4. Interdisciplinary foundations for HCI developments

In analysing the learning curve of HCI the companion paper identified the fifth-generation era commencing in 1980 as marking a transition from empiricism to theory. The fourth generation was certainly one of empirical rules for dialog engineering based

on experience rather than theory, and the following section briefly traces its development. The current fifth generation should be one of theoretical principles giving generative foundations for dialog engineering, and the final major section traces the sources of these foundations.

DIALOG RULES IN THE FOURTH GENERATION

As experience in the design and use of interactive systems grew, various authors presented guidelines for the design of effective human-computer dialog: Hansen (1971); Wasserman (1973); Martin (1973); Kennedy (1974); Gaines & Facey (1975); Pew & Rollins (1975); Cheriton (1976); Gilb & Weinberg (1977); Gaines (1978); Gebhardt & Stellmacher (1978); Turoff, Whitescarver & Hiltz (1978); Gaines (1981); Gaines & Shaw (1983, 1984*a*). Shneiderman (1980) has collected many of these rules together, and Maguire (1982) has analysed some of them systematically, noting contradictions and proposing their resolution. Reference works have also been based on the rules such as the short, practical book of Kidd (1982) and the very detailed review of Williges & Williges (1984).

The number of guidelines for dialog engineering has grown over the years. Ours started as a set of 11 rules for *programming interactive dialog* in 1975, growing to 15 in 1978, 17 in 1981, 22 as a *human protocol* in 1983, and 30 *proverbs* in our 1984 book on the *Art of Computer Conversation.* 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 to over 500, 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, indicates that the time is ripe for a new approach to dialog engineering.

THE MOVE TOWARDS THEORETICAL FOUNDATIONS

The hope for the fifth generation is that the theoretical foundations of HCI will become sufficiently secure in their own right to break away from the current paradigm, in which we cope with the technology, to one in which we engineer HCI in its own right and mould the technology as a result. This requires major advances in our understanding of communication at the level of the human mind and these will be just as applicable to person-person communication as they are to human-computer communication. We can already see the convergence of computational linguistics, artificial intelligence, cognitive science and HCI, to some deeper understanding of symbolic communication amongst human, and human-like, entities.

The companion paper indicated some of the links between AI and HCI, and it is interesting to attempt to characterize the different influences that these have on problem solving. Abstract definitions are probably of less value than a simple example that exaggerates the differences in approach. Consider a problem of overcoming the barriers to database access by casual users:

—the AI approach to this problem might be to build a better database system with understanding of the user, his requirements, and natural language communication,

i.e. to put all the load on the computer and make it clever enough to deal with the casual user:

—the applied psychology approach might be to develop a training program for casual users that gives them as rapidly and effectively as possible the skills to use the database, i.e. to put all the load on the person and make him clever enough to cope with the computer;

—the HCI approach might be to determine where the problems lie in the interface between user and computer and design a communication package that helps him formulate his requests in a way natural to him, but which can be translated easily into a form natural to the database, i.e. to remove as much of the load as possible from both systems and share what is left between them.

In practice, a pragmatic system designer will take what he can get from all three areas and put together a working system. However, the example shows, in very simplistic terms, both the close relationships between the disciplines and their differing orientations.

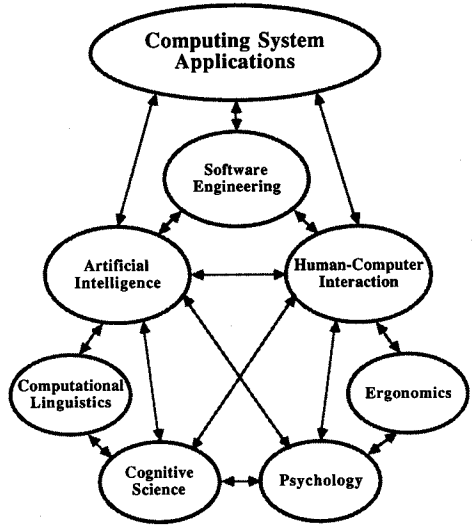

FIG. 2. Influences between the disciplines underlying fifth- and sixth-generation computing-system applications.

Figure 2 incorporates related disciplines into an influence diagram indicating the rich structure now underpinning human-computer applications. AI and HCI are now major influences on computer applications and both have influenced software engineering: AI by the development of symbolic languages for knowledge processing and HCI by developments for dialog engineering. AI has also had a direct influence on HCI by the introduction of intelligent interfaces (Rissland, 1984) and natural-language processing. It has also had an indirect influence through its promotion of interest in cognitive science and computational linguistics which provide tools for HCI. These areas have also become influential in psychology and ergonomics providing foundations for HCI. Completing the loop, psychology in its turn has influenced thinking in AI where the goal has been to emulate human intelligence rather than create truly *artificial* intelligence.

FIFTH-GENERATION HCI OBJECTIVES

What we might reasonably ask for at this stage in the development of HCI is a set of *principles* that systematically *generates* rules for dialog engineering. The principles should be *grounded* in system theory, computer science and cognitive psychology. The principles should be *applicable* to the entire range of possible dialog styles and technologies, now and in the future. The principles should be *operational* so that they can be embedded in standard procedures, dialog shells widely available for all interactive systems.

There have been a number of developments that satisfy some of these objectives. Thimbleby (1980) has explored a number of generative principles for dialog design such as *dialog determination.* Nelson (1980) has proposed *virtuality* as an analytic principle, noting that we project to the user a virtual world from within the computer. Shneiderman (1983) has subsumed a range of such recent developments in display-oriented systems as examples of *direct manipulation.* Shackel (1983) has classified them also using the concept of *usability.* Norman (1984) has defined *levels* of HCI. Shaw & Gaines (1984) have analysed the role of computing as a development in the *media* of communication. Each of these concepts gives a logic for HCI design that is supported psychologically and operationally, and hence satisfies some of the objectives expressed above.

The concept of a *virtual machine* has proved an extremely useful one in computer science (Gaines, 1975; Gaines & Shaw, 1982; Weeganaar, 1978). Its extension from the hardware and software to the programmer and user allows a range of HCI studies to be set in an integrated framework (Gaines, 1979). We can view any computer system as either a programmable machine with certain features together with a program, or as a (programmable) virtual machine with extended features (Gaines & Shaw 1985, Shaw & Gaines 1985*a*). As noted above, the user sees the virtual machine not the underlying one, and it is this virtual "reality" that underlies the user interface in machines such as the Star and Macintosh.

The algebraic analysis of the virtual machine 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 (Weimer & Palermo, 1974) such as Scandura's (1973) structural learning. The use of computational models derived from artificial intelligence studies is particularly promising (Bobrow & Collins 1975; Findler 1979; Gentner & Stevens 1983), as are those models based on psychological principles with systematic foundations such as personal construct psychology (Shaw, 1980, 1981; Gaines & Shaw, 1981, 1984*c*). The explanation of problems of learning structure from behavior in terms of complexity-theoretic measures (Gaines, 1976*a*, *b*) suggests that they may be valid as a measure of psychological difficulty in understanding virtual machines.

Embedding HCI within the virtual machine hierarchy gives it a theoretical framework that can both be formalized and linked to many practical consequences and related disciplines. Further research on this approach to the analysis of HCI problems and principles would be valuable in grounding HCI within the framework of software engineering. The design principles that we apply to HCI are part of a larger context of studies of applied psychology, human and computer communication, and general systemic principles. Although they may require fresh validation in the context circumstances of HCI, they have richer origins and support than studies of HCI alone. This section explores the wider context systematically.

5. Systematic analysis of person–computer interaction

The basic situation of person-computer interaction may be shown diagrammatically as in Fig. 3. The primary analysis has to consider: the nature of a person; the nature of a computer; the interface from a person to a computer; the interface from a computer to a person; and the interaction between person and computer through these interfaces.

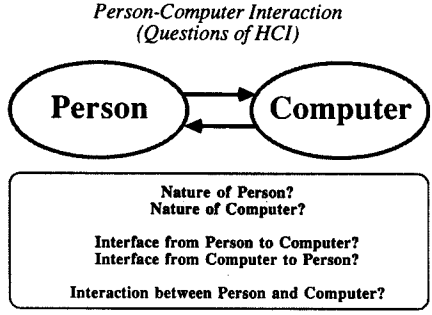

FIG. 3. Analysis of person-computer interaction.

SYSTEM-SYSTEM INTERACTION

There is also the possibility of analysis through generalization, noting that people and computers are both examples of *systems* in general as shown in Fig. 4. At an abstract level, one notes that systems can interact in many ways and that 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

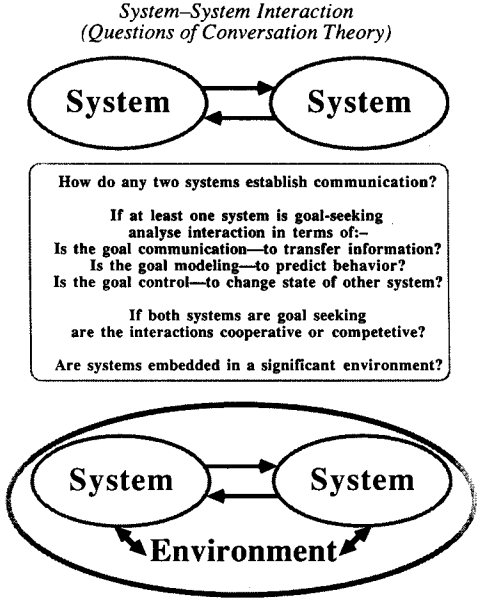

FIG. 4. Analysis of system-system interaction.

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. It is also possible to consider goals relating to the coupled systems that arise outside them and may not be manifest to them. In this context, we may wish to consider the possible interaction of one or both systems with an environment as shown at the bottom of Fig. 4.

There are many system-theoretic principles that have been developed for the cases of Fig. 4 which are instantiated in the situation where one system is a person and the other a computer. Wiener (1948) emphasized this in his development of *cybernetics* as the study of "communication and control in men and machines", and many principles of communication theory and control theory apply directly to HCI. For example, the dialog rule that *the user should dominate the computer* (Gaines & Shaw, 1983) derives from a stability-theoretic result in control theory, that two coupled systems with similar time constants may oscillate unstably around their intended equilibrium state: 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 relationship between dialog rules and their system-theoretic foundations is one of dialectical tension. Concrete rules are needed that can be applied within a range of contexts without excessive development and inference. However, when the context varies, the rules 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 enquiry without ill effects (Witten, Greenberg & Cleary 1983).

The theory of system-system interaction most directly relevant to HCI is Pask's (1975) *conversation theory* which considers the problem of systems validating mutual communication and of observers comprehending its occurrence. Pask gives systemic criteria for communication to occur and has used these as a basis for designing person-computer dialogs in education through computer-based learning. Conversation theory has been applied by Coombs & Alty (1984) to the design of expert systems, by Shaw (1984) to knowledge engineering, and by Begg (1984) to the analysis of HCI in CAD systems. It provides basic systemic principles that may be used in the context of specific situations to generate rules of dialog design for effective HCI.

INFORMATION THEORETIC PRINCIPLES OF SYSTEM-SYSTEM INTERACTION

There are some basic information-theoretic principles applying to system-system interaction that clearly define some of the boundaries to aspirations in HCI. Consider the three levels of complexity of communication between two systems:

Level 1: the message itself;
Level 2: the system affected by that message;
Level 3: the identification of the system.

A message can be very simple at level 1 yet have massive effects on the system receiving it. There can be a high complexity gain, e.g. in the President giving the communication which initiates a nuclear attack. A message may need to be extremely complex to have a small effect on a system with complex dynamics. The message may have to initiate an action and then cope with its unwanted consequences by initiating further actions and coping with their consequences until the effect is focused as required. Thus there is no necessary relationship between complexity at level 1 and complexity at level 2.

In general, the sender of a message need not understand all the effects on the recipient to make the message passing process useful. The President could take the action with only the objective of retaliation without knowing all the consequences. Thus there is no necessary relationship between the level 2 of the sender and that of the receiver.

If the range of messages possible is to be sufficient to drive all the dynamics of the receiver then the complexity of level 1 must match that of the receiver's level 2. If the sender is to generate all the messages possible then the complexity of the sender's level 2 must match that of level 1. However, it is possible to match complexity without matching structure and hence even in these circumstances there is only equivalence in complexity between sender and receiver.

Consider the situation, however, in which the sender wishes to understand the effect of the message on the receiver. The sender now has to have available a model of the receiver which is similar in structure to the receiver. The availability of this model may not be problematic if the sender and receiver are analogous such that the sender itself is a model of the receiver. Thus understanding between complex systems may be unproblematic and not invoke complexity at level 3 if sender and receiver are sufficiently alike. However, if the two systems are not alike, then the sender needs an alternative source from itself of a model of the receiver. This may consist of other analogs of the receiver and hence again not invoke level 3.

If the sender has to build a model of the receiver then a new level of complexity emerges. The complexity of a system able to create a model of another is necessarily much greater than that of the system being modeled. The information necessary to model a system is very much greater than that necessary to pass messages between them. If two dissimilar systems are attempting to establish communication based on mutual understanding, then we have a paradox. On the above argument, each must be more complex than the other in order to achieve this. Hence complete understanding is impossible between dissimilar systems. This is a reasonable result. In practice, an adequate level of mutual understanding to achieve certain goals may involve only a very small part of the two systems' structures. They may each restrict their complexity to that which is necessary to cope with the goals.

These considerations show that we have to be very careful in the use of the notion of *complexity* in HCI. It is a relative term. We also have to beware making assumptions that problems at level 3 can be solved. It is an operational definition of a *natural* system for a user that it avoids such problems.

COMPUTER-SYSTEM INTERACTION

If one of the systems in Fig. 4 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 as shown in Fig. 5. The design principles applicable to computer-equipment interfaces

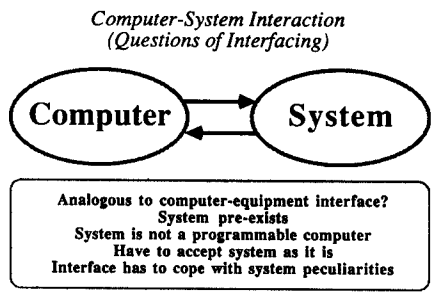

FIG. 5. Analysis of computer-system interaction.

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 take it as it is and design an interface that copes with its peculiarities. The dialog rule *use the user's model* (Gaines & Shaw, 1983) derives from this, 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 and 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, miskeying giving rise to errors in the other direction, and so on. The dialog rule *validate data on entry* (Gaines & Shaw, 1983) is a principle of communication over a noisy channel.

COMPUTER-COMPUTER INTERACTION

When the systems on both sides of Fig. 4 are computers as shown in Fig. 6, then it is possible to define *protocols* that it is reasonable to expect any programmable digital system to be able to implement. The *Open System Interconnection* (OSI) ISO standard

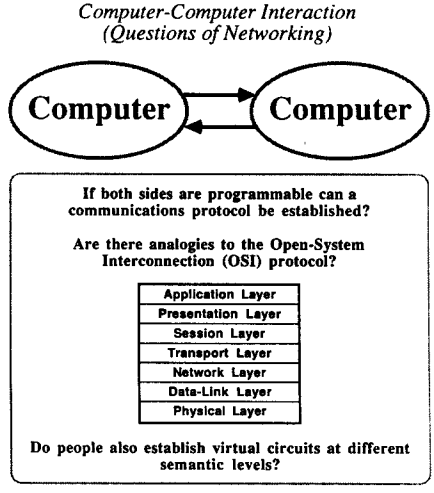

FIG. 6. Analysis of computer-computer interaction.

(Day & Zimmerman, 1983) is particularly interesting because it hierarchically structures computer-computer protocols for networks in a way that may have relevance for person-computer protocols. For example, dialog engineering rules may be seen as defining a *human protocol* (Shaw & Gaines, 1983).

The concept of an *open system* is 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 (Gaines, 1983). 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.

As shown in Fig. 6, 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 audiovisual perceptual processes, and there is much to be gained in applying the OSI concepts to the human protocol. Taylor has applied such protocol concepts to the analysis of person-computer dialog as a hierarchical communications system (Taylor *et al.* 1984).

PERSON-SYSTEM INTERACTION

When one of the systems is specialized to be a person we have the classic case of man-machine interaction as shown in Fig. 7. Considerations of people interacting with

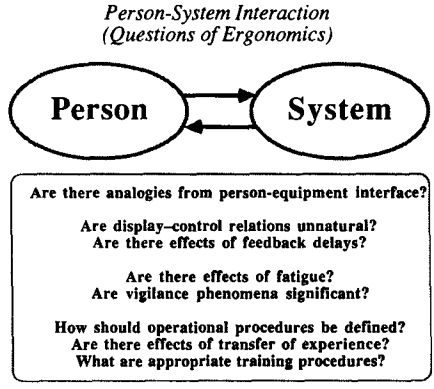

FIG. 7. Analysis of person-system interaction.

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 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 as shown in Fig. 8. Modern linguistic theory (Bennett, 1976) has become increasingly concerned

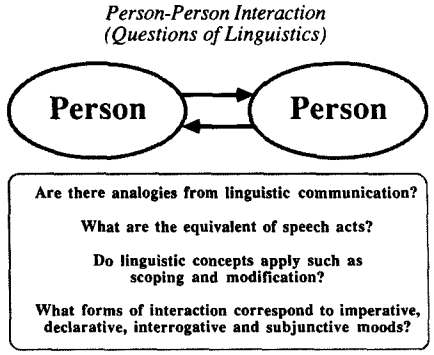

FIG. 8. Analysis of person-person interaction.

with the interaction between participants in a dialog, rather than a view of linguistic output as a predefined stream to be decoded. This provides a rich source of models for person-computer interaction, particularly as AI techniques take us closer to emulating people and their language behavior. There are also useful analogies of casual users in transactional analysis of the behavior of strangers meeting—*What do you say after you say hello*? (Berne, 1974). Taylor notes that many of the principles of language such as scoping and pronomial reference are general principles of communication that occur in graphic/gesture dialog with computers that involves no textual interaction (Taylor *et al.*, 1984).

SYSTEMATIC ANALYSIS OF DIALOG SHELL DESIGN

Each of the five situations discussed provides a range of principles for interaction which are either directly applicable to HCI or, through analogies, have indirect application. One reason we have been able to develop effective interactive systems is that we can draw upon experience of these related situations. We can also use them more systematically to give foundations for dialog engineering. For example, consider the hierarchy of levels of dialog rule generation (Gaines & Shaw, 1984*b*) shown in Fig. 9: systemic principles; particular features of people and computers; consideration 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* (Gaines & Shaw, 1983) 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 (Gaines, 1976*a*), and we

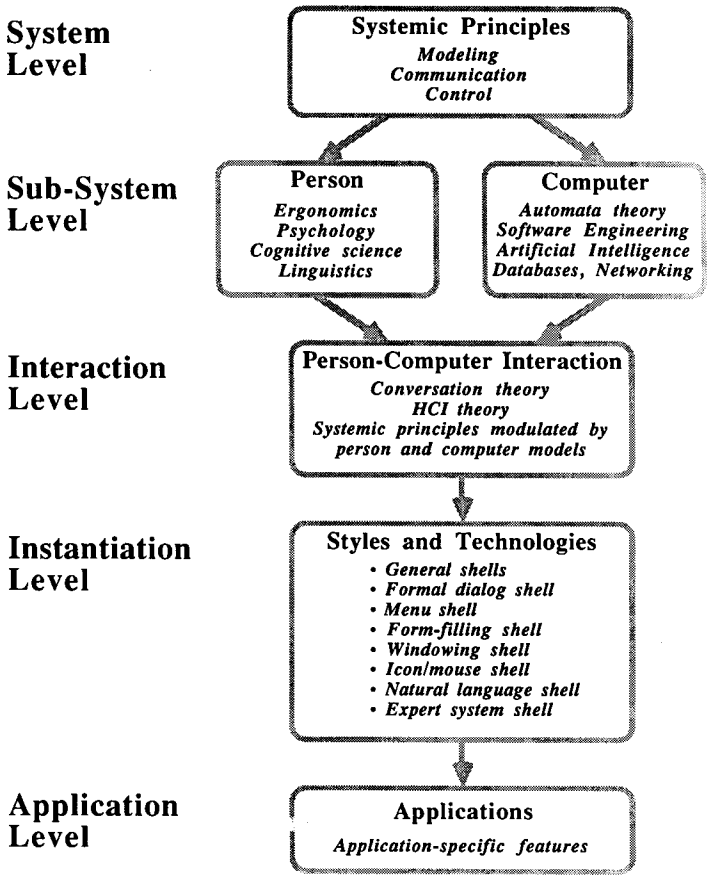

FIG. 9. The hierarchy of levels underlying dialog shell design.

would not regard the rule as significant unless we knew 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 this 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. It is possible to define shells which incorporate many of the formal dialog rules for prompt-response (Gaines & Shaw, 1984*a*), and shells for form-filling transaction processing and menu-based systems are in widespread use. The Xerox Star user interface was an early example of such a shell and recent "window" systems provide shells for personal computers (Gordon, 1984). INTELLECT and ASK are examples of natural language shells for database access. Several commercial companies have introduced window shells for the IBM PC and Apple Macintosh which has a general icon/mouse shell. Expert system shells such as EMYCIN and AL/X (Gevarter, 1983) are recent developments that incorporate significant new dialog techniques such as the uniform availability of *why* and *how* queries.

6. Conclusions: *Quo vadis*?

This paper and its companion have analysed the historical development of HCI, its place in the infrastructure of computing, its relations to hardware and software engineering, and to artificial intelligence, the current state of the art in dialog styles and engineering, and the rich interdisciplinary sources of foundations for the development of HCI theory during the fifth generation. A variety of styles of dialog has been identified and illustrated and it has been noted that, although they follow a developmental sequence, they do not necessarily outmode one another and many possible approaches may be taken to solving the same HCI problems. The influences of other disciplines on HCI and their contribution to software engineering has been examined. A rich environment for HCI studies, theory and applications now exists. This has been explored systematically by analysing the various analogies to human–computer interaction possible when the parties are taken to be general systems, people or computers. In particular, many principles underlying HCI are shown to have systemic foundations independent of the nature of the systems taking part and these may be analysed control-theoretically and information-theoretically. Each of the five situations discussed provides a range of principles for interaction which are either directly applicable to HCI or, through analogies, have indirect application.

It has been suggested that one reason we have been able to develop effective interactive systems is that we can draw upon experience of these related situations. We can also use them more systematically to give foundations for dialog engineering. However, we cannot wait for an adequate model of the brain, a complete human psychology, or even comprehensive empirical studies of human factors in HCI, to build complex human–computer systems. Thus, in looking at where we are going a more pragmatic viewpoint is useful also. Norman (1983) has provided this viewpoint by suggesting four strategies for providing design principles for HCI which can be summarized as:

Strategy 1: make system designers more aware of the significance of HCI;
Strategy 2: provide methods and guidelines for HCI design;
Strategy 3: provide software tools for HCI design;
Strategy 4: separate HCI design from other design tasks.

As we have shown, the first strategy has been an ongoing effort since the first generation of computers, became of major commercial significance in the fourth and fifth generations, and can now be regarded as well established if not yet totally 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 (Woodmansee, 1984).

The second strategy was the major thrust of HCI studies through the fourth generation, but it has been so effective in generating guidelines as to become ineffective in promoting their use. When several hundred rules for effective HCI are grouped together, they become daunting and the user begins to ask for them to be condensed or automated.

The third strategy is that most apparent in the fifth generation and has resulted in the provision of tools for HCI design such as Moran's (1981) *command language grammar*, Jacob's (1983) *executable specifications*, and Edmonds & Guest's (1984) *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* as discussed above. 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.

This is the situation today, nearly forty years after the birth of computers. We know much about the human-computer interface. We have many examples of good HCI. We still also 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 that we provided foundations of dialog engineering for the human-computer interface as explicit and well-founded as those for hardware and software engineering.

For the future, we suggest a fifth strategy that takes into account three major requirements in advancing the state of the art:

—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;
—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;
—such theories must contain adequate models of people *and* computers, and of their interaction.

The fifth strategy is:

Strategy 5: provide an overall framework for HCI in systems involving multiple people in various roles and multiple computer subsystems 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.

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 inseparable 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 subsystems 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. It is necessary to have models of both subsystems

that can be used together in a way that accounts naturally for their interaction, for the reflection of the user model needed in the computer and for the reflection of the computer that is the user's model. We have emphasized the need to combine systemic, psychological and computational models in the design of dialog shells. The need is even greater in considering total system design.

This paper has been influenced by discussions over many years and we are grateful to Ray Chapman, Walter Doherty, Peter Facey, Martin McGuire, Ben Shneiderman and Harold Thimbleby. We are particularly grateful for recent interaction with Jack Edwards and Martin Taylor that has sharpened many of the arguments. Financial assistance for this work has been made available by the National Sciences and Engineering Research Council of Canada.

References

BAWDEN, A., GREENBLATT, R., HOLLOWAY, J., KNIGHT, T., MOON, D. & WEINREB, D. (1979). The Lisp machine. In WINSTON, P. H. & BROWN, R. H. Eds, *Artificial Intelligence: an MIT Perspective*, pp. 343-373. Cambridge, Massachusetts: MIT Press.

BEGG, V. (1984). *Making Computer Aided Design Tools More Useable: a Study of a Complex Task Shared by People and Machines.* London: Kogan Page.

BENNETT, J. (1976). *Linguistic Behaviour.* Cambridge, U.K.: Cambridge University Press.

BERNE, E. (1974). *What Do You After You Say Hello?* London: Andre Deutsch.

BEWLEY, W. L., ROBERTS, T. L., SHROIT, D. & VERPLANK, W. L. (1983). Human factors testing in the design of Xerox's 8010 "Star" Office Workstation. In JANDA, A., Ed., *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 72-77 (December). Boston.

BOBROW, D. G. & COLLINS, A., Eds (1975). *Representation and Understanding: Studies in Cognitive Science.* New York: Academic Press.

CHERITON, D. R. (1976). Man-machine interface design for time-sharing systems. *Proceedings of the ACM National Conference*, pp. 362-380.

COOMBS, M. & ALTY, J. (1984). Expert systems: an alternative paradigm. *International Journal of Man-Machine Studies*, **20**, 21-43.

DAVIS, R. & LENAT, D. B. (1982). *Knowledge-Based Systems in Artificial Intelligence.* New York: McGraw-Hill.

DAY, J. D. & ZIMMERMAN, H. (1983). The OSI reference model. *Proceedings IEEE*, **71**, pp. 1334-1340 (December).

EDMONDS, E. A. & GUEST, S. P. (1984). The SYNICS2 user interface manager. *Interact '84: First IFIP Conference on Human-Computer Interaction*, **1**, 53-56. Amsterdam: Elsevier.

FINDLER, N., Ed. (1979). *Associative Networks: Representation of Knowledge by Computers.* New York: Academic Press.

GAINES, B. R. (1975). Analogy categories, virtual machines and structured programming. In GOOS, G. & HARTMANIS, J., Eds, *GI-5. Jahrestagung. Lecture Notes in Computer Science*, **34**, pp. 691-699. Berlin: Springer Verlag.

GAINES, B. R. (1976*a*). On the complexity of causal models. *IEEE Transactions on Systems, Man & Cybernetics, SMC-6(1)*, pp. 56-59 (January).

GAINES, B. R. (1976*b*). Behaviour/structure transformations under uncertainty. *International Journal of Man-Machine Studies*, **8**, 337-365.

GAINES, B. R. (1978). Programming interactive dialogue. In *Pragmatic Programming and Sensible Software*, pp. 305-320. Uxbridge, Middlesex, U.K.: Online Conferences.

GAINES, B. R. (1979). The role of the behavioural sciences in programming. *Structured Software Development*, Vol. 2. pp. 57-68. Maidenhead, U.K.: Infotech International.

GAINES, B. R. (1981). The technology of interaction-dialogue programming rules. *International Journal of Man-Machine Studies*, **14**, 133-150.

GAINES, B. R. (1983). From word processing to image processing in office systems. *Proceedings of International Electrical, Electronics Conference and Exposition, IEEE 83CH1955-4*, pp. 622-625.

GAINES, B. R. & FACEY, P. V. (1975). Some experience in interactive system development and application. *Proceedings IEEE*, **63**, 155-169 (June).

GAINES, B. R. & SHAW, M. L. G. (1981). A programme for the development of a systems methodology of knowledge and action. In RECKMEYER, W. J., Ed., *General Systems Research and Design: Precursors and Futures*, pp. 255-264. Louisville, Kentucky: Society for General Systems Research (January).

GAINES, B. R. & SHAW, M. L. G. (1982). Analysing analogy. In TRAPPL, R., RICCIARDI, L. & PASK, G., Eds, *Progress in Cybernetics and Systems Research, Vol. IX*, pp. 379-386. Washington: Hemisphere.

GAINES, B. R. & SHAW, M. L. G. (1983). Dialog engineering. In SIME, M. & COOMBS, M. J., Eds, *Designing for Human-Computer Interaction*, pp. 23-53. London: Academic Press.

GAINES, B. R. & SHAW, M. L. G. (1984*a*). *The Art of Computer Conversation: a New Medium for Communication.* New Jersey: Prentice Hall.

GAINES, B. R. & SHAW, M. L. G. (1984*b*). Dialog shell design. *Proceedings of INTERACT '84: 1st IFIP Conference on Human-Computer Interaction*, pp. 344-349. Amsterdam: North-Holland.

GAINES, B. R. & SHAW, M. L. G. (1984*c*). Foundations of expert systems. In KACPRZYK, J. & YAGER, R. R., Eds, *Management Decision Support Systems Using Fuzzy Sets and Possibility Theory.* Rheinland: ISR Interdisciplinary Systems Research, Verlag TUV.

GAINES, B. R. & SHAW, M. L. G. (1985*a*). Foundations of human-computer interaction. *Proceedings of CIPS/ACI CONGRESS 85*, pp. 384-388. Montreal: Canadian Information Processing Society (June).

GAINES, B. R. & SHAW, M. L. G. (1986). From timesharing to the sixth generation: the development of human-computer interaction. Part I. *International Journal of Man-Machine Studies*, **24**, 1-27.

GEBHARDT, F. & STELLMACHER, I. (1978). Design criteria for documentation retrieval languages. *Journal of the American Society for Information Science*, **29**, 191-199.

GENTNER, D. & STEVENS, A. L., Eds (1983). *Mental Models.* Hillsdale, New Jersey: Lawrence Erlbaum.

GEVARTER, W. B. (1983). Expert systems: limited but powerful. *IEEE Spectrum*, **18**, 39-45.

GILB, T. & WEINBERG, G. M. (1977). *Humanized Input.* Cambridge, Massachusetts, U.S.A.: Winthrop Publishers.

GORDON, P. (1984). What is integrated software? *PC World*, **2**(11), 72-77 (October).

HANSEN, W. J. (1971). User engineering principles for interactive systems. *Proceedings of the Fall Joint Computer Conference*, **39**, pp. 523-532. New Jersey: AFIPS Press.

HARRIS, L. R. (1977). User oriented data base query with the ROBOT natural language query system. *International Journal of Man-Machine Studies*, **9**, 697-713.

HARRIS, L. R. (1984). Experience with INTELLECT. *AI Magazine*, **5**(2), 43-50.

HILL, I. D. (1983). Natural language versus computer language. In SIME, M. & COOMBS, M. J., Eds, *Designing for Human-Computer Interaction*, pp. 55-72. London: Academic Press.

JACOB, R. J. K. (1983). Executable specifications for a human-computer. In JANDA, A., Ed., *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 28-34 (December). Boston.

KENNEDY, T. C. S. (1974). The design of interactive procedures of man-machine communication. *International Journal of Man-Machine Studies*, **6**, 309-334.

KIDD, A. (1982). *Man-Machine Dialogue Design.* BTRL, Ipswich, UK: Martlesham Consultancy Services.

MAGUIRE, M. (1982). An evaluation of published recommendations on the design of man-computer dialogues. *International Journal of Man-Machine Studies*, **16**, 237-261.

MANUEL, T. (1983). Lisp and Prolog machines are proliferating. *Electronics*, **56**(22), 132-137 (November).

MARTIN, J. (1973). Design of Man-Computer Dialogues. New Jersey, USA: Prentice-Hall.

MORAN, T. P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, **15**, 3-50.

MOTO-OKA, T., Ed. (1982). *Fifth Generation Computer Systems.* Amsterdam: North-Holland.

MYERS, W. (1982). Lisp machines displayed at AI conference. *Computer*, **15**(11), 79-82 (November).

NELSON, T. (1980). Interactive systems and the design of virtuality. *Creative Computing*, **6**(11), 56-62 (November).

NORMAN, D. A. (1983). Design principles for human-computer interfaces. In JANDA, A., Ed., *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 28-34 (December). Boston.

NORMAN, D. A. (1984). Stages and levels in human-machine interaction. *International Journal of Man-Machine Studies*, **21**, 365-375.

OGDEN, W. C. & BROOKS, S. R. (1983). Query languages for the casual user: exploring the middle ground between formal and natural languages. In JANDA, A., Ed., *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 161-165 (December). Boston.

PASK, G. (1975). *Conversation, Cognition and Learning.* Amsterdam: Elsevier.

PASK, G. (1980). Developments in conversation theory—Part I. *International Journal of Man-Machine Studies*, **13**, 357-411.

PEW, R. W. & ROLLINS, A. M. (1975). Dialog specification procedure. *Report No. 3129.* Cambridge, Massachusetts: Bolt, Beranek & Newman.

RISSLAND, E. L. (1984). Ingredients of intelligent user interfaces. *International Journal of Man-Machine Studies*, **21**, 377-388.

ROSENBERG, J. (1983). A feature approach to command names. In JANDA, A., Ed., *Human Factors in Computing Systems: CHI '83 Conference Proceedings*, pp. 116-119 (December). Boston.

ROSENBERG, J. & MORAN, T. P. (1984). Generic commands. *Proceedings of INTERACT '84: 1st IFIP Conference on Human-Computer Interaction*, 1, pp. 360-364. Amsterdam: Elsevier Science Publishers.

SCANDURA, J. M. (1973). *Structural Learning. I. Theory and Research.* New York: Gordon & Breach.

SHACKEL, B. (1983). Usability: its meaning and evaluation in human-computer system. *Tutorial 9 at CHI '83.* Boston: ACM SIGCHI.

SHAW, M. L. G. (1980). *On Becoming a Personal Scientist.* London: Academic Press.

SHAW, M. L. G., Ed. (1981). *Recent Advances in Personal Construct Technology.* London: Academic Press.

SHAW, M. L. G. (1984). Knowledge engineering for expert systems. *Proceedings of INTERACT '84: 1st IFIP Conference on Human-Computer Interaction*, pp. 328-332. Amsterdam: North-Holland.

SHAW, M. L. G. & GAINES, B. R. (1983). Does the human component in a network have a protocol? *Proceedings of International Electrical, Electronics Conference and Exposition, IEEE 83CH1955-4*, pp. 546-549.

SHAW, M. L. G. & GAINES, B. R. (1984). Fifth generation computing as the next stage of a new medium. *Proceedings of National Computer Conference*, **53**, pp. 445-451. Arlington, Virginia: AFIPS Press.

SHAW, M. L. G. & GAINES, B. R. (1985*a*). Systemic principles for human-computer interaction. In BANATHY, B. H., Ed., *Systems Inquiring: Theory, Philosophy, Methodology*, pp. 377-386. Seaside, California: Intersystems Publications.

SHAW, M. L. G. & GAINES, B. R. (1985*b*). Developments in dialog engineering. *Proceedings of National Computer Conference*, **54**, pp. 493-499. Arlington, Virginia: AFIPS Press.

SHNEIDERMAN, B. (1980). *Software Psychology.* Cambridge, Massachusetts: Winthrop.

SHNEIDERMAN, B. (1983). Direct manipulation: a step beyond programming languages. *Computer*, **16**(8), 57-69 (August).

SMITH, D. C. (1977). *Pygmalion.* Basel: Birkhauser.

SMITH, D. C. (1983). Visibility in the Star user interface. *Proceedings of Graphics Interface '83*, p. 181. Ottawa: National Research Council of Canada.

SMITH, D. C., IRBY, C., KIMBALL, R., VERPLANK, B. & HARSLEM, E. (1983). Designing the Star user interface. In DEGANO, P. & SANDEWALL, E., Eds, *Integrated Interactive Computing Systems*, pp. 297-313. Amsterdam: North-Holland.

STA (1985). *Promotion of R & D on Electronics and Information Systems That May Complement or Substitute for Human Intelligence.* Tokyo: Science and Technology Agency.

TAYLOR, M. M., MCCANN, C. A. & TUORI, M. T. (1984). The interactive spatial information system. *DCIEM Report 84-R-22.* Toronto: DCIEM.

THIMBLEBY, H. (1980). Dialogue determination. *International Journal of Man-Machine Studies*, **13**, 295-304.

THOMPSON, B. H., THOMPSON, F. B. & HO, T.-P. (1983). Knowledgeable contexts for user interaction. *Proceedings of National Computer Conference*, **52**, pp. 29-38. Arlington, Virginia: AFIPS Press.

TUROFF, M., WHITESCARVER, J. & HILTZ, S. R. (1978). The human-machine interface in a computerized conferencing environment. *Proceedings of the IEE Conference on Interactive Systems, Man and Cybernetics*, pp. 145-157.

WASSERMAN, T. (1973). The design of idiot-proof interactive systems. *Proceedings of the National Computer Conference*, **42**, pp. M34-M38. New Jersey: AFIPS Press.

WEEGENAAR, H. J. (1978). Virtuality and other things like that. *Proceedings of IEEE COMCON, CH1388-8/78*, pp. 287-293.

WEIMER, W. B. & PALERMO, D. S., Eds (1974). *Cognition and the Symbolic Process.* Hillsdale, New Jersey: Lawrence Erlbaum.

WEIZENBAUM, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Journal of the ACM*, **9**, 36-45.

WIENER, N. (1948). *Cybernetics.* Cambridge, Massachusetts: MIT Press.

WILLIGES, B. H. & WILLIGES, R. C. (1984). Dialogue design considerations for interactive computer systems. MUCKLER, F. A., Ed, *Human Factors Review: 1984*, pp. 167-208. Santa Monica, California: Human Factors Society.

WINOGRAD, T. (1972). *Understanding Natural Language.* Edinburgh: Edinburgh University Press.

WITTEN, I. H., GREENBERG, S. & CLEARY, J. (1983). Personalizable directories: a case study in automatic user modelling. *Proceedings of Graphics Interface '83*, pp. 183-189. Ottawa: National Research Council of Canada.

WOODMANSEE, G. H. (1984). The VisiOn experience—from concept to marketplace. *Proceedings of INTERACT '84: 1st IFIP Conference on Human-Computer Interaction*, **2**, pp. 162-166. Amsterdam: Elsevier.

ZLOOF, M. M. (1982). Office-by-example: a business language that unifies data and word processing and electronic mail. *IBM Systems Journal*, **21**, 272-304.