Paper for **Compcon 82**, September 20-24, Washington DC.

THE HUMAN PROTOCOL

by Brian R. Gaines and Mildred L. G. Shaw

Centre for Man-Computer Studies,
92, Shakespeare Tower, Barbican, London EC2, UK.

Department of Computer Science, York University,
Downsview, Toronto, Ontario, Canada M3J 1P3.

Abstract

In most networks some of the nodes interface to people, and protocols are required for effective communication across this interface. Some aspects of such protocols have been studied in the past as part of **dialog engineering**. This paper presents the results of such studies systematically as the basis for defining an effective **human protocol**.

I. Introduction

At some nodes in any network, protocols are required for interaction with people. People-computer protocols have some similarities with computer-computer protocols and analogous rules apply to both at a systemic level. However, people-computer protocols need also to take into account characteristics of people that are generally not found in interaction with equipment -- notably that a person is adaptive and explores and models his or her environment. This paper presents rules for engineering effective dialog with people that have been developed over many years of experience with a wide range of interactive systems [4, 5, 7, 8, 9] and compares and contrasts them with the protocols for communication between purely technical systems. It is argued that people-computer protocols can be developed and standardized and that this should be seen as an important part of network technology.

This paper has been structured deliberately as a concise set of 'rules' with only brief explanation so that the protocol can be seen clearly. The basis of the rules and links to the literature are discussed in detail in [9] and general background is available in [1, 10, 11, 12, 16, 17].

II. Basis for People-Computer Protocols

One interesting feature of people is that they generally do not have manuals which define the appropriate protocols to be used in operating with them. One way of obtaining this information is to ask them how they would like the interaction to be structured. However, many of them come to a network for the first time with no experience of interaction with a computer system and hence their answers may be misleading. The first rule in designing a protocol is to avoid this problem.

Rule 1 Introduce through experience: Interactive systems need to be experienced before their operation can be discussed meaningfully. Get prospective users onto a terminal on a related, or model, system before discussing their preferred protocol for interaction with their own system.

As well as mistrusting the computer-naive user's view of a suitable protocol we also need to distrust the application-naive system analyst's view. The terminology, presentation and sequence of dialog should reflect the user's model of the activity.

Rule 2 Use the users' model: Base the protocol on the users' model of their activities so that the interactive dialog is similar to a conversation between two users mutually accepting this model.

People are continually changing their modes of operation, particularly in settling down after the perturbation of the introduction of a new system. Hence the first two rules above need continuous application.

Rule 3 Design never ceases: Use the flexibility of the system through programming so as to close the adaptive loop between system and user through

yourself as designer.

This can be done most effectively through the network's data collection capabilities.

Rule 4 Log activities: Use the network to maintain selective records of system and user activities and provide programs to analyse these in terms of, for example, errors, broken down by user and by dialog sequence.

III. Problems of User Adaption

The human ability to model and adapt, and their continuous application of it, causes new problems for the protocol designer.

Rule 5 The user will model the system: Do not assume that the user is a passive static system to be controlled, modelled and directed by the computer. Evaluate the protocol in terms of its effect on an actively changing user who is attempting to comprehend the system.

One major problem for the user in modelling the system is random behavior on its part, such as delays in a timeshared system.

Rule 6 Avoid acausality: Make the activity of the system a clear consequence of the user's actions.

One important feature of a system to aid a user modelling it is consistency.

Rule 7 Consistency: Ensure that all terminology and operational procedures are consistently applied throughout the protocol.

A complementary feature to this in aiding modelling is uniformity, not just that features remain the same but that they are always available.

Rule 8 Uniformity: Ensure that all terminology and operational procedures are uniformly available throughout the protocol.

IV. Person or Computer in Control

There is a difficult choice to be made in many situations as to who should have primary control over the interaction. It is tempting to try to analyse the user characteristics and take them into account automatically, e.g. by switching to a more supportive protocol when naive user behavior is detected. However, the information to do this accurately is not easy to obtain. Either the computer or the user should dominate the interaction or there will be instability. If the computer is to dominate it must be programmed to, and have sufficient information to, model the user. If the user is to dominate then the computer system must be simple to understand.

Rule 9 User should dominate computer: At the present state of the art the user should dominate the system.

A simple way of enabling the skilled user to adapt the system himself is to allow grouping of responses.

Rule 10 Parallel-sequential tradeoff: Allow the user maximum flexibility to make his responses holistically (in parallel) or serially (in sequence) according to his wishes.

In enabling the user to control the interaction it is important that he is kept aware of the state of the protocol.

Rule 11 Make the state of the dialog observable: give the user feedback as to the state of the dialog by making an immediate unambiguous response to any of his inputs which may cause the dialog to branch - the response should be sufficient to identify the type of activity taking place.

Another important feature in enabling the user to cope with the system is to give him access to its protocols where they apply.

Rule 12 Query in context: Distribute information and tutorial material appropriately throughout the protocol to be accessed by the user through a simple uniform mechanism, e.g. responding with a "?".

Responses to queries should not be verbose, but additional information should be available if required.

Rule 13 Query in depth: Organize the response to queries so that the user accesses brief memory aids first at any point but has further access to more detailed explanations.

The documentation of the protocol should enable the user to put himself in the context of interacting with the system.

Rule 14 User manuals should be based on actual user dialog: Illustrate the use of the protocol by showing actual dialog sequences that achieve specific objectives. Illustrate the structure of the system through dialog sequences where queries are used to elicit information about the protocol.

A protocol complying with the rules given here should be its own best training aid.

Rule 15 Train through experience: get users interacting with an actual system as their initial training and introduce them to the system facilities such as query-in-depth.

V. Error Detection and Recovery

Error detection and recovery procedures are just as important with people-computer protocols as with computer-computer protocols. They should take into account the person's need to explore the system characteristics and to be able, and to feel able, to recover readily from any consequences of this.

Rule 16 Validate data on entry: Check syntax, and values against norms.

However, norms that filter errors also may reject actual data.

Rule 17 Be flexible in validation: Beware of rejecting data or querying too much as being outside norms. Allow the user to over-ride the error checking where appropriate.

Always allow also for undetected errors and give the user back control at events with major consequences.

Rule 18 Have the user himself revalidate major updates: Show the consequences of major actions clearly before acting upon them and ask for user confirmation.

When a user detects errors he should be able to exit without problems. The user should be able at any stage in a transaction to abort it cleanly with a system command that takes him back to a well-defined checkpoint as if the transaction had never been initiated.

Rule 19 Provide a reset command: that cleanly aborts the current activity back to a convenient checkpoint.

For minor entry errors the abort and restart is wasteful and a simpler mechanism is desirable.

Rule 20 Provide a backtrack facility: that allows a user to return through the dialog sequence in reverse.

Errors in entry are minimized by providing default values representing the most likely entry.

Rule 21 Provide default values: that are accepted by the user by, for example, returning a null message.

This is also useful for record update and correction.

Rule 22 Make corrections through re-entry: use the entry dialog with default field printouts from a record as a means of correcting the record.

VI. Synopsis of the Rules

The 'rules' for a human protocol presented here are quite clearly at different conceptual levels. However, they can be organized within a coherent framework by noting that they reflect three sources of information open to the systems analyst in designing an effective person-computer protocol.

The rules of Section I are concerned with the user as one source of information, and with the operational experience of the system as another. This is shown as the loop of **user -- analyst -- design -- operation -- experience** in Figure 1, with both user and analyst changing their system models through operational experience and the analyst also obtaining direct feedback from the user.

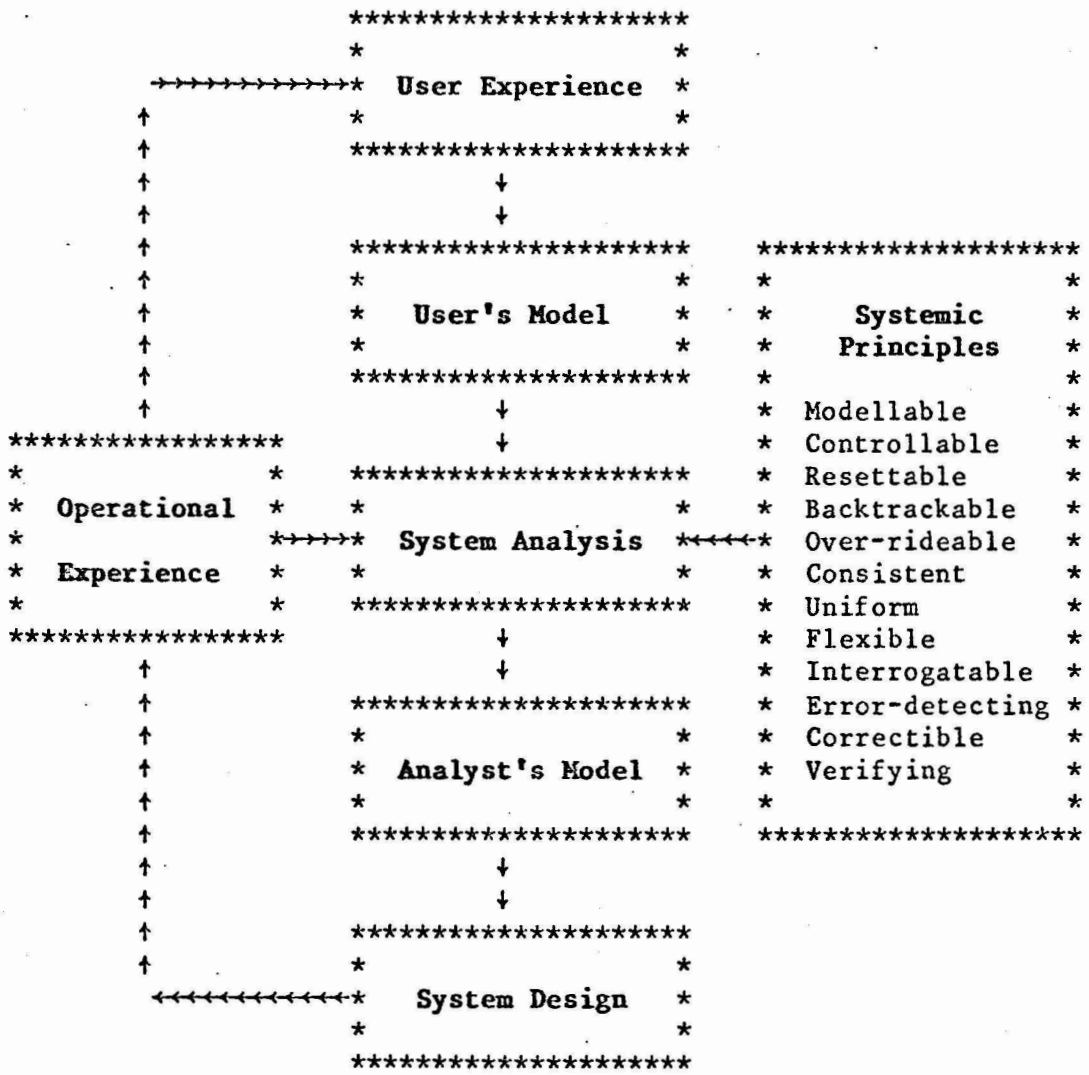

Figure 1 Three Sources for System Analysis of the Human Protocol

The rules of Sections II through V are **systemic** in nature and apply not to particular systems and users but to the human protocol in general. They are a third source of information about the system design available to the systems analyst and apply regardless of the particular system requirements. These rules can be summarized in human terms as the system being **supportive** through being easy to understand, observe and control. The rules formalize the psychological concept of being supportive into systemic concepts underlying a suitable protocol.

VII. Implementing the Protocol

Implementing the human protocol defined above is best done through a standard dialog procedure which is written once and used thereafter as the only means of interacting with the user. The main programming language feature necessary for such a procedure is the capability to handle variable length strings as data items and particularly as parameters to procedures. Languages such as Snobol and Lisp are ideal in the facilities they provide but even Fortran and Pascal are generally provided with some form of primitive string data type and processing facilities. BASIC has good string handling facilities but very poor procedural modularity with its lack of parameter passing and local variables.

Actual commercial systems obeying the rules described above have been written in the interactive systems programming language Basys II [2, 3, 6] which provides special facilities for string processing. These are based on compiler-compiler language constructs for string analysis incorporated in a BASIC-like language that is extended to have procedure calls with parameters and local variables. The standard Basys II operating system has a dialog processing protocol embedded in it and all dialog, including that of the operating system job control language, goes through this protocol.

The protocol procedure is called with parameters giving the prompt string, query-in-depth replies, and branches for reset and backtrack. Its operation is most readily understood by considering how each rule is implemented in turn:

Rules 1, 2 and 3 do not require any special features in the procedure. Rule 2, **Use the user's model**, implies ease of of decoding strings returned by the procedure in an arbitrary format natural to the user, and this is particularly simple in Basys II. We have used the interactive computer itself as a tool to derive the user's model [13, 14, 15]. Rule 3, **Design never ceases**, implies some form of interactive programming language with a modular structure and again Basys II satisfies this requirement. However, other languages offer equally good string handling or can be provided with suitable procedures.

Rule 4, **Log activities**, is the first where the dialog procedure itself should provide facilities. One of the parameters that can be passed to it is the name of a log-file procedure. This may default either to no action or to logging the complete dialog. However, it is also possible by this mechanism to provide selective logging of the dialog at points which require study or are dependent on other parameters such as the identity of the user.

Rules 5, 6 and 9, **The user will model the system, Avoid acausality,** and **User should dominate computer,** in the context of the dialog protocol are warnings not to make it too "clever" in its actions.

Rules 7 and 8, **Consistency** and **Uniformity,** are primarily ensured by requiring all dialog to go through the standard procedure.

Rule 10, **Parallel-sequential tradeoff,** is readily implemented. The string returned by the dialog procedure is in a global variable and the calling routine extracts what items it wants from it. When the dialog procedure is next called it checks the global variable and only if it is null does it go to the user for input. Thus the user may input several items at the same time and they will be passed to the appropriate callers of the dialog procedure.

Rules 12 and 13, **Query in context** and **Query in depth,** are implemented by passing to the procedure a table of strings which are used to reply to each successive question-mark.

Rules 14 through 18, on **Manuals, Training and Validation,** depend on how the procedure is used rather than its own characteristics.

Rule 19, **Provide a reset command,** is implemented through the procedure returning not to its normal address but to the branch passed as the reset parameter. This branch can usually be set up as a default for a dialog sequence and should be to code which cleans up any actions taken during the course of the dialog before returning to the top-level program.

Rule 20, **Provide a backtrack facility,** is implemented in a similar way to rule 19 except that the abnormal branch is to the parameter passed which will usually be the previous call on the dialog procedure or any parameter calculations immediately preceeding it.

Rules 21 and 22, **Provide default values** and **Make corrections through re-entry,** are implemented by passing to the dialog procedure a computed prompt string which contains the initial or previous value of the item requested.

Thus most of the key rules of the human protocol may be implemented simply and automatically through a dialog procedure. It is useful also to incorporate in the procedure facilities for diagnostic purposes and for taking the input from a command file rather than a terminal.

VIII. Conclusions

This paper has proposed an approach to formalizing the **human protocol** through **dialog engineering** which has been used in a wide range of interactive systems in the last fifteen years. It has proved very successful in allowing naive users to cope rapidly and adequately with complex interactions whilst not frustrating highly skilled users. The rules proposed are intended as guidelines rather than absolutes and each is substantiated by arguments based on both theory and practice. Some aspects of the rules are specific to the particular style of dialog discussed but they are basically systemic and general to all forms of interaction. We hope that this presentation will not only act as a

guide to those attempting to create effective interactive systems but also as a stimulus to others to improve, vary, and generally develop both the rules and the entire domain of dialog engineering.

In particular we hope that this paper brings into the domain of study of protocols for distributed systems the human factors concepts necessary to effective system operation. The person is not just another system component and cannot, and should not, be treated as such. However, there is much in common at the systemic level between those aspects of protocols necessary for effective computer-computer communication and those for effective person-computer communication. The **human protocol** can be treated systematically, formally and effectively - that is the message of this paper.

IX. References

1. Coombs,M.J. & Alty,J.L. (eds.), **Computing Skills and the User Interface.** Academic Press, London, 1981.

2. Gaines,B.R., Interpretive kernels for microcomputer software. In (eds.), **Proceedings Symposium Microprocessors at Work,** Soc. Electronic & Radio Technicians, University of Sussex, UK, 1976, 56-69.

3. Gaines,B.R., A mixed-code approach to commercial microcomputer applications. **Microprocessors in Automation and Communications,** 1978, IERE Conference Proceedings No.41, 291-301.

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

5. Gaines,B.R. & Facey,P.V., Some experience in interactive system development and application. **Proceedings IEEE,** 1975, 63, 155-169.

6. Gaines,B.R. & Facey,P.V., BASYS - a language for processing interaction. **Proceedings Conference Computer Systems and Technology, (IERE No.36),** University of Sussex, UK, 1977, 251-262.

7. Gaines,B.R., Facey,P.V. & Sams,J.B.S., An on-line fixed-interest investment analysis and dealing system. **Proceedings European Computing Congress, EUROCOMP 74,** 1974, 155-169.

8. Gaines,B.R., Facey,P.V. & Sams,J.B.S., Minicomputers in security dealing. **Computer,** 1976, 9, 6-15.

9. Gaines,B.R. & Shaw,M.L.G., Dialog engineering. In Sime,M. & Coombs,M.J. (eds.), **Designing for Human-Computer Communication.** Academic Press, London, 1982, to appear.

10. Gilb,T. & Weinberg,G.M., **Humanized Input.** Winthrop Publishers, Cambridge, Mass., USA, 1977.

11. Guedj,R.A., ten Hagen,P.J.W., Hopgood,F.R.A., Tucker,H.A. & Duce,D.A. (eds.), **Methodology of Interaction.** North-Holland, Amsterdam, 1980.

12. Martin,J., **Design of Man-Computer Dialogues.** In (eds.), Prentice-Hall, New Jersey, USA, 1973.

13. Shaw,M.L.G., **On Becoming a Personal Scientist.** Academic Press, London, 1980.

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

15. Shaw,M.L.G. & Gaines,B.R., Eliciting the real problem. In Wedde,H. (ed.), **International Working Conference on Model Realism,** Bonn, Federal Republic of Germany, 1982.

16. Shneiderman,B., **Software Psychology.** Winthrop, Cambridge, Massachusetts, 1980.

17. Smith,H.T. & Green,T.R. (eds.), **Human Interaction with Computers.** Academic Press, London, 1980.