ZOG: A Man-Machine Communication Philosophy

G. Robertson, A. Newell, and K. Ramakrishna

Department of Computer Science
Carnegie-Mellon University

August 4, 1977

ZOG is a rapid response, large network, menu selection system used for man-machine communication. The philosophy behind this system was first developed by the PROMIS (Problem Oriented Medical Information System) Laboratory of the University of Vermont. ZOG will be used in a number of task domains to help explore and evaluate the limits and potential benefits of the communication philosophy. This paper discusses the basic ideas in ZOG and describes the architecture of a system that has just been implemented to carry out that exploration and evaluation.

This work was supported by the Office of Naval Research under contract no. N00014-76-0874. It was also partially supported by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

1. Introduction

We have hardly begun to understand how to communicate effectively with computers. In part, the problem is one of evolution. As the computer continues to evolve into more powerful forms, the resident systems become more extensive and intelligent, requiring and supporting more sophisticated dialogs, while simultaneously the power available to the act of communication itself increases. Both requirements and opportunities are continually new. In part, the problem is one of analysis. Only a few studies have explored the man-computer interface. There certainly is no well developed theoretical framework for evaluating interfaces or understanding how they might be improved. In part, the problem is one of invention. We only slowly get ideas for new ways to communicate. Mostly we have been so bound by technology that the task has seemed one of removing the obvious glitches and annoyances of existing interfaces and getting the band-width up to some comfortable level, all within bounds of economy.

What would be an ideal communication medium between ourselves and machines? No one knows. We have a tendency to reach in two directions for the answer. One is reaction to present interfaces. Yesterday it was faster card readers and faster printers. Today, it is 2400 baud terminals with a modicum of "intelligence" (meaning local computing power) and good interactive command languages. The other source is communication with our fellow man. We wish to speak and listen, using natural language. This seems ideal because it is the best we know now, and the product of immense biological and social evolution. We also wish to paint and sketch, drawing on another area where expressive skills have long natural development.

If we can draw any lesson from the development of computers it is that we should seize on any notion that seems to expand the frontiers of the possible -- that offers to open up our horizons. Most such efforts will be still born, not liberating or not technologically feasible. Even then, if done with some scientific curiosity and attention, they can leave a residue that will help later on. Occasionally, however, such probes can set the stage for new developments.

We report here on the beginnings of a probe. We describe a particular interface for man-computer interaction that seems to us to have quite novel properties. We call our version of the scheme ZOG (which stands for nothing, but is short, easily pronounced and easily remembered). The basic idea is not ours. It is the development of the PROMIS group at the University of Vermont medical school, who originated the idea, designed and implemented a complete system incorporating it, and brought it to practical use in a major way.[1] The scheme itself is easily stated. Communication is via

---

[1] /Our debt to the PROMIS system will be apparent throughout. It has provided the impetus, the guidance and the evidence to date for the type of system we are considering. We would like to make a more personal acknowledgment to the members of the PROMIS group for their help and encouragement, and especially to

menu selection on terminals with display capabilities. Selection is instantaneous and is accomplished by touching areas of the display screen associated with each option. The result of selection is a frame with further selections (with knowledge and action provided in passing). The network of such frames is large enough so that all communication is by this means; in practice this means very large indeed. Thus, communication from man to computer is by discrete selection of semantically meaningful options, from computer to man by visual display of natural language text within a structured format.

Menu selection being a common technique in man-machine communication (Martin, 1973), the potentially revolutionary character of the PROMIS system is not easily appreciated directly from a short written description. We will go into that issue shortly. Before we do, let us briefly set out the present ZOG project and the contents of this report.

The PROMIS interface system is embedded within a total system, called the Problem Oriented Medical Information System, which is the main focus of concern of the PROMIS group. Though they have had a system running for almost five years now its impact on the development of man-machine interfaces generally has been miniscule. We do not know of another system with the same essential features. We were sensitized to their work from an attempt of our own in 1972 to produce a similar system (the original ZOG, this current one is ZOG2). We decided to attempt to extract the scheme from its habitat in the PROMIS application so as to study and exploit it as a general communication interface. Our goal is to find out whether this interface does indeed have the potential it appears to have, to demonstrate it, and to study its parameters in order to understand and optimize it.

This paper is our report on the initial work toward this objective, which is development of a basic system with which to work. In the first part of the paper we introduce the basic ideas, and describe and illustrate the way ZOG works. In the second part we describe the system architecture of ZOG. This includes a description of the design and implementation of the current version, called ZOG2, which runs on a PDP10, but does not yet provide support for rapid response and large networks. It also includes a description of the design for ZOG on C.mmp, which will provide rapid response and large network support. Finally, we discuss the potential of this philosophy, its cost, and some thoughts about the next steps to take. The appendices provide detailed summaries of the design specifications for the current system and a list of terms used throughout this report.

Jan Schultz and Steve Cantrill. Two of us (GR and AN) have served on a Technical Advisory Committee to the National Center for Health Services Research of HEW in connection with PROMIS (and GR still serves in that capacity). Our membership in that committee was triggered by our earlier attempts at ZOG1, which were put aside in part because of a failure to adopt some critical aspects of the PROMIS user interface, as described herein.

2. Basic Ideas in ZOG

Let us first describe the ZOG system and how it operates. The user faces a terminal which is displaying a frame. Figure 1 shows a typical ZOG frame. There is text at the top, a list of options below the text, a column of pads at the right side, an area called a workspace below the options, and a horizontal line of pads at the very bottom. The user, at his discretion, selects one of the options or pads. Suppose he selects option 3. Immediately, the frame on the display is replaced by a new one with all the same parts: text, options, vertical column of pads, horizontal line of pads and workspace. Most of the content will be new except for the horizontal pads, which provide a continuously available set of search and help functions. For example, selecting the "back" pad (b) will cause a redisplay of the frame of Figure 1. Selecting some pads or options will cause various actions to happen. If the user had selected the PRINT pad, the textual information on this display would have been printed on the printer.

```
General background on ZOG                                          ZOG3

ZOG is part of a research effort to understand communication between humans
and computers.  Various aspects of this research effort are described below.

   1. System specifications of ZOG

   2. What is ZOG used for -- functional characterization

   3. Scientific issues behind ZOG

   4. Who is doing ZOG? Where? When? What sponsors?

   5. Prior research and antecedents

   6. Examples of ZOG projects (real and in progress)

   7. Developing your own ZOG net                                  P-PRINT


a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 1. Typical ZOG Frame

That is all there is to ZOG as far as external mechanics are concerned. The user traverses a sequence of frames of his own selection, acquiring the information therein and taking the actions offered to him. It stands, at this level, simply as a menu selection scheme, distinguished only by its ability to take actions in addition to present knowledge (a property shared by many other menu selection schemes). Later in the

paper a more extended example will be given of its operation, but it will genuinely be more of the same.

A preliminary notion of what ZOG can be used for is also needed. Like any general purpose interactive programming language, it can serve in any communicative capacity whatsoever: command language, data base retrieval system, CAI system, guidance system, interrogation system, question-answering system, etc. Also, like any programming language, what it is good for, as opposed to what it can conceivably be used for, is not determined by gross structure, but by more subtle features of operation. However, it is important to take ZOG as being used not only for initial guidance or with novice users, but also for skilled operation; and not only for exploring knowledge bases, but also for taking action.

In PROMIS, such a system is used as the sole interface in accomplishing the total set of hospital functions on a ward: keeping patient medical records, taking patient histories directly from patients, prescribing drugs and treatments, monitoring patient progress, checking treatments for side reactions, and retrieving medical knowledge. It is used by doctors, patients, nurses, paramedics, and administrative people. It performs a full range of communicative functions with users who range widely in sophistication and in direct skill with the system. The communication interface is only a part of the total system that accomplishes all these functions, but it is a central one.

We can now enumerate the additional basic features of ZOG and fill in the missing design specifications. The centrality of the features varies, as does the amount of evidence for their role. But together they define the ZOG2 system. We list them in Figure 2 and then discuss each of them.

1. Rapid response. Upon selection the new frame appears instantly.

2. Simple selecting. The user's act of making a selection is a simple unitary gesture.

3. Large network. The network of frames is large enough to accommodate all communication and knowledge-exchange with the user.

4. Frame simplicity. The frame display is simple enough to be easily and quickly assimilated.

5. Transparency. The entire system is completely open and understandable to the user.

6. Communication agent. The system is usable with existing programming systems to provide guidance in how to use the programs and how to interpret their results.

7. Subnet facilities. There exists a hierarchical data organization for networks.

8. Personalization. The user can modify and augment the network to suit himself.

9. External definition. An external data format exists that completely defines a ZOGNET.

10. Uniform search. A uniform scheme exists for searching and orienting in the network.

Figure 2. Basic Features of ZOG2

2.1. Rapid Response

Upon selection the new frame appears instantly. Instantly is defined with respect to the user. It means fast enough so the user feels the flow of frames to be limited only by his own volition. If traversing a highly familiar network (as in

specifying some operand to be worked on), then he can move through the frames almost as a single skilled gesture. If he wishes to take a quick glimpse of a next unknown frame, he feels no hesitation because he need only wait for two responses (one down, one back).

How fast "instantly" must be in seconds is not fully known. PROMIS operates with .25 seconds 70% of the time. It is not likely to be much slower than this. ZOG is targeted for .05 seconds 70% of the time in order to permit exploration of this parameter.

2.2. Simple Selecting

The user's act of making a selection is a simple unitary gesture. The time it takes the user to make a selection acts in series with the response of the system; it must be equally rapid. Its speed has two aspects: learning what the response should be (since new selections are always occurring) and executing the response. PROMIS uses a touch screen which solves both these problems: the user simply touches the display at the local area where the option is stated. ZOG also uses a touch screen. In addition it uses a single character selection from a keyboard.

2.3. Large Network

The network of frames is large enough to accommodate all communication and knowledge-exchange with the user. Large enough is again defined with respect to the user. It means that at every frame there are options to be taken that deal with whatever information, help, elaboration and explanation is required. The options lead to other frames which also provide whatever is necessary, and so recursively. The user finds himself in a world where all of his questions and all of the data he requires have already been laid out in advance in the network of frames -- where his needs have been anticipated. (An important exception to this will be taken up in Section 2.6.)

How many frames is "large enough" is not known. The total system of frames constitutes a finite state system. Considering the combinatorial nature of life, the conclusion could be that finite networks are incapable of satisfying this requirement for any interesting task. (Recall Chomsky's (1957) early stated view that no finite state grammar can adequately portray a natural language.) Without doubt this design feature leads to large networks. The PROMIS network appeared to satisfy this requirement with about 30,000 frames; it may eventually grow to 100,000 frames. As with the criteria for rapid response, perfection is not necessary (i.e., the user need not remain within the frame system 100% of the time). In PROMIS the user types in a response rather than selecting an option (indicating the absence of an appropriate precoded response) about 1% of the time. Not much is known about the details of this criteria or good measurements of it. What is known is that PROMIS has produced one

system that clearly is adequate, in a general enough environment to engender some optimism.

2.4. Frame Simplicity

The frame display is simple enough to be easily and quickly assimilated. The power of the technique comes from the fractionation of the total task into small communicative pieces with control by the user of which of these he wishes to acquire. If each frame were, say, like a textbook page, then substantial assimilation would be required, and the user would be thrown back on his own scanning and organizing resources without any help from ZOG. The natural criteria is that the user should never have to acquire knowledge other than what constitutes his final solution or what is necessary to find this solution (by the nature of the task, not of ZOG).

What is "simple enough" is relative to the user. Even less is known about it or how to measure it than about rapid system response or large networks. Perhaps it can be measured structurally by amounts of text and options; perhaps by the average residency time per frame; perhaps it requires knowing exactly what knowledge was absorbed. PROMIS frame design, which has evolved over several years, tends toward a few sentences of text and no more than half a dozen options.

This design constraint is common to all menu selection systems and does not seem unique to ZOG. It is useful to stress it, however, because it implies even larger networks, that would otherwise be the case. Faced with the large network problem, a natural engineering inclination is to permit the knowledge per frame to increase. This principle inhibits that inclination.

2.5. Transparency

The entire system is completely open and understandable to the user. It should seem totally open to him exactly why the system is doing what it is doing and what it takes to obtain more information from it or to get it to do something. It should appear completely controllable and non-mysterious. The effect is stated in terms of user's perceptions, since what counts is the way the user reacts to it. The ideal user's model should be that of the perfect instrument.

The specifications already laid down go a long way toward meeting the requirement of transparency: menu selection, rapid response, large network and simple frames. This creates a structure that is simple in concept and completely under the user's control. However, within these constraints it would be easy to realize obscure networks. Thus, the burden of this specification falls on the details of network and frame-content design. ZOG, like any programming system, creates an architecture within which one write programs. The programs for ZOG are networks of filled-in frames. Thus this requirement can be taken as a requirement on programming style.

Exactly how to spell out this style in concrete terms is not known. The net-building experience from PROMIS has not been well articulated and no attempts have been made to measure this, however crudely. A large number of frames have been built in PROMIS and have undergone substantial polishing so that the impression of the system tends to accord moderately well with this view. Furthermore, some expected consequences of achieving this requirement have occurred in PROMIS; for example, patients on admission use the system without any training to take their own background histories.

2.6. Communication Agent

The system is usable with existing programming systems to provide guidance in how to use the programs and how to interpret their results. This is not a requirement that derives from PROMIS, but from our earlier attempt, called ZOG1, to build a similar system. PROMIS is an active system in that the selections can execute programs. But it is a closed system implemented on dedicated hardware. An entire world of applications is excluded if all programs for ZOG must be coded specially for it. It then becomes a particular form of interactive programming language, though with many special features. But if it can somehow be integrated with any existing programming system, then it becomes a much more flexible tool.

This requirement is realized by making ZOG be a communication agent. That is, ZOG sits astride the communication path between the user and a program, called the subjob, where it can monitor, interpret and modify the input and output streams in both directions. There is a communication language associated with ZOG itself, but it does not operate as a regular programming language. Rather, it translates the user's selections into messages to send to the subjob. It also monitors the subjob's output to the user and can translate these into new messages to the user or into selections, which cause new displays to be shown to the user.

This language will be described in detail in the section on the architecture; it is simple in concept and implementation. The essential point here is that ZOG can operate as a front-end for any program whatsoever. It can provide explanation, instruction and guidance in working with a system. Likewise it can interpret the output for the user and suggest what to do. To do any of these things, of course, requires a ZOG program (i.e., a network) specifically designed to know about the subjob. How good a job it does depends on the extent and sophistication of the network.

An alternative way of describing this structure is that ZOG is a command language. It is through ZOG that the user makes contact with all the resources of a computer, executes programs, and does his housekeeping (file management, message handling, etc.) Its usefulness as a command language, of course, arises not from any special logical character, but in its potential for explaining and guiding.

The use of ZOG as a communication agent necessarily carries with it the potential to violate transparency. ZOG clearly need not know all about the programs

for which it fronts. In general, it cannot do so (e.g., if permitting access to another programming language such as ALGOL68 or LISP). Thus, the user will generally see a mixed system in which he will be aware when the subjob is running and he is no longer dealing with ZOG.

The ability to have a system that can interpose between the user and any other job running on the host computer requires certain capabilities in the resident operating system within which the communication agent and the other job run. Such capabilities exist with the TOPS-10 operating system on the PDP10 and the HYDRA operating system on C.mmp, the two computers on which ZOG will run, but such capabilities are not necessarily available in all operating systems.

2.7. Subnet Facilities

There is a hierarchical data organization for networks. This organization is called the subnet and it operates essentially like a subroutine. Subnets have names and can be handled as units. The user can orient himself by subnets, always getting back to the top of a subnet. There can be subnets within subnets recursively.

Given modern practice in data structures, there is nothing striking about this requirement. Nevertheless, it is important for the creation and modification of networks, and for the efficient implementation of large networks. Its importance for the user's operation in the net is less clear and there are no strong grounds for insisting on it (rather than simply having the user see the network as one large integral structure).

2.8. Personalization

The user can modify and augment the network to suit himself. The goal of transparency by itself probably requires that a user be able to make small changes easily to any existing network, representing his own understanding and preferred way of dealing with the material of the net. There are some closely related additional reasons, such as increasing efficiency. But, as with the communication agent requirement, additional worlds of application become possible if networks are easily constructed and modified. Users can extend explanations, not just to clarify what is there, but in an application centered around the growth of ZOG (e.g., representation of the issue-network of some problem). Full retrieval systems, involving both augmentation and access, become possible, as opposed simply to searches of fixed data bases.

The requirement for easy modification and augmentation implies that an editor for frames and nets become an integral part of the system. It is called ZED (for ZOG Editor). There would be a requirement for an editor in any event, but if it were viewed as just for some class of professional "net builders" its facilities and its integration into

ZOG could be seen as much less crucial. In the current scheme, the alter mode part of ZED is evokeable as a selection from any frame.

2.9. External Definition

An external data format exists that completely defines a ZOGNET. Because frames are display structures there is a temptation to define them only as internal data structures. In fact this was true of the initial PROMIS system (but not of the current iteration), and we expect it is true of most menu selection systems. However, considerations of ultimate portability require that some machine independent definition of a ZOGNET exist.

This requirement takes on the status of a system imperative as soon as large networks are contemplated. A network of 50,000 frames represents a great body of knowledge (like a 2000 page text book), and becomes extremely valuable. It is easier to get a new ZOG system up and running on a new computer than it is to create 50,000 frames. Under these conditions, portability of the frame library is mandatory.

There is an interaction between this requirement and the specification of ZOG to be a communication agent. An alternative path is for ZOG to have associated with it a full programming language. This leads to large amounts of the knowledge in a ZOGNET being encoded in procedures in this language. Because the operating environment of this programming language is the display, it is much less perspicuous than most standard higher level languages. Documentation of these procedures so that the network becomes in fact portable and maintainable is a serious issue (and this is proving so with the current PROMIS system). The ZOG communication language is quite limited and simple (e.g., it has no conditional or loop control), so that essentially all of the knowledge in the net is encoded in the surface structure of frames and their connections. Thus it turns out that a simple data form involving strings of alphanumeric text is quite adequate.

The solution adopted is to embed our simple format within a bibliographic system which is in public use at CMU, called BH (Newcomer, 1976). BH provides the requisite data handling and printing facilities. It is also oriented towards relatively large files. It is described in the section on system architecture. The point to be made here is that the system has a complete external definition simple enough to permit exportation.

2.10. Uniform Search

A uniform scheme exists for searching and orienting in the network. Transparency requires that the user find the system extremely easy to understand and deal with, even when working in networks new to him (which will often occur in acquiring bodies of new knowledge). The operational aspects of how the user finds his

way in the net are as important as the simplicity of the frames themselves. The incipient difficulty is that the frame provides only a small and local view on the world so that the user must move around in the net to acquire knowledge, leaving all but the current frame out of sight.

The solution to this would appear to take the form of a set of operations and conventions for how any net is structured, which the user may rely upon and become familiar with. We have formulated a set of principles and conventions that seem reasonable based on the experience of PROMIS and our own limited explorations. But there is little data to support the particular structure.

The uniform search requirement is a constraint on network design, on the content of frames and their arrangement. No features of the architecture proper reflect it. As a programming system, ZOG is defined and usable independent of this requirement, just as a computer is defined without specifying an operating system or an assembler. However, just as with an operating system, ZOG cannot be run without something that provides tools for searching through the net. Thus, we add this requirement here, even though it is a pure ZOG-software requirement.

The solution adopted in ZOG attempts to satisfy the following principles:

1. No sudden death. No selection taken by a user produces a change that is irreversible. This applies both to movements in the network and to actions taken. Where this is not possible explicit confirmation will be required.

2. Standardized pads. There is a set of selections with standard names that are available in all frames. (This is the horizontal line of pads consisting of: alter, back, display, help, mark, next, return, zog, and exit.)

3. Anchor points. It is always possible to return to known frames that play the role of anchor points. Anchor points should be dynamically determinable. (This is realized in part by the pads: back, mark, return and zog.)

We have described the basic ideas that we are exploring. Now, let us examine an extended example to see what the existing system looks like from the user's point of view.

3. Sample Interaction

Figures 3 to 14 show a typical initial interaction. We are able to fit two frames into a single page of the report with a line of commentary below each. Several critical aspects of the ZOG design can not be illustrated by this means, in particular the rapid response and large network. But it should make the format of ZOG and its style of operation more concrete.

We will postulate a visitor to CMU who wishes to find out about research on production systems. Our visitor has never used ZOG or the computer before. He is shown a terminal (which is in the monitor state on the CMUA system), and someone leans over his shoulder and types "ZOG" <carriage return> for him. The frame in Figure 3 comes up on the screen, and the user is on his own.

```
        ZOG - An Interactive Guide System                          ZOG1

Welcome to ZOG.
You are at the starting place for the entire ZOG network.
ZOG is a guidance system which will help you understand some
subjects and help you to use some programs and systems.

You use ZOG by selecting the options it offers.
Type h (for help) to find out how to use ZOG
or just explore by typing the character in front of any option.
Typing z (for ZOG) will always bring you back here.

  1. Table of contents

  2. Utilities for ZOG frame building

  3. Sending a message to the designers of ZOG

  4. Clear frame backup list

  5. Go to an arbitrary frame - prompts for number

a-alter  b-back  d-display  h-help                  r-return        ↑C-exit
```

Figure 3. User enters ZOG. He selects '1' (Option 1).

```
                    Table of Contents                                ZOG2

The following subnets are available in this version of ZOG.

  1. General background on ZOG

  2. ZED -- The frame editor

  3. Computing facilities

  4. People

  5. Projects

  6. Education

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 4. User wishes to know about a research project. He selects '5'.

```
Projects in the Computer Science department                Proj1     Proj1

Research activity in the department is organized and conducted as 'projects'.
Projects may be large, or small, group, or individual - the usual use of the
term denotes a group of people who meet regularly, discuss and work on some
common system or problems.
Current projects are described below under broad subject classifications.

  1. Artificial Intelligence

  2. Programming languages

  3. Operating Systems

  4. Computer system design

  5. Computer system evaluation

  6. Data-base design and evaluation

  7. (Other projects)

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 5. User is interested in an AI project. He selects '1'.

```
Artificial Intelligence - Projects                    Proj1    Proj2

The AI community here comprises a number of people in Computer Science,
Electrical Engineering, and Psychology departments at CMU.  Typically,
the project is the work of a small number of people, though a few large
projects occupy a large number of people.
The projects are listed below, some large ones first, but otherwise not very
systematically.

  1. The Speech Understanding System project (HARPY, HEARSAY)

  2. The Vision project

  3. The Production Systems Working Group (IPS, OPS)

  4. Automated Mathematician (AM)

  5. Heuristic search (Chess, CAPS)

  6. Evaluation of problem-solving techniques

  7. (Other projects)

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 6. User wants to know about Production Systems. He selects '3'.

```
The Production Systems Workshop                       Proj1    Proj11

The Production Systems group consists of people from Computer Science and
Psychology who are studying the problem of building and evaluating the
performance of a large production system (the Instructable PS) capable
of functioning in a complex, and unpredictable task environment.

  1. What are Production Systems?

  2. The people in the group

  3. Instructable Production System (IPS)

  4. Papers and technical reports published

  5. Organizational and procedural issues (meetings etc.)

  6. Other groups working with Production Systems


a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 7. User wants a definition. He selects '1'.

```
What are Production Systems?                                    Proj1    Proj16

Production Systems are a species of abstract computing devices, which have
certain interesting properties from the viewpoint of cognitive psychology,
application of problem-solving techniques, and problem-representation issues.

  1. Abstract computing devices - general theory

  2. Production Systems as a computing device

  3. Production Systems in problem-solving

  4. Production Systems in cognitive psychology

  5. Production System implementations

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 8. User selects 'n' (which should give him Option 2 from Figure 7).

```
The people in the Production Systems Workshop                   Proj1    Proj17

The group consists of the people listed below.  Some particular
concerns or responsibilities are indicated.
Dr. Newell is on sabbatical and Mike Rychener is the formal leader of
the group through the summer of 1977.

  1. Allen Newell: Adaptation of methods to new uses

  2. Mike Rychener: Design and construction of the Kernel for IPS

  3. John McDermott: Design and maintenance of the Task Environment (OTE)

  4. Charles Forgy: Design and construction of the Official Production System

  5. Pat Langley: Induction from the TE

  6. Kamesh Ramakrishna: Theories of problem-solving

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 9. User decides to go back. He selects 'b'.

```
The Production Systems Workshop                                  Proj1    Proj11

The Production Systems group consists of people from Computer Science and
Psychology who are studying the problem of building and evaluating the
performance of a large production system (the Instructable PS) capable
of functioning in a complex, and unpredictable task environment.

  1. What are Production Systems?

  2. The people in the group

  3. Instructable Production System (IPS)

  4. Papers and technical reports published

  5. Organizational and procedural issues (meetings etc.)

  6. Other groups working with Production Systems


a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 10. User wants to know when the group meets. He selects '5'.

```
Organizational and Procedural issues                             Proj1    Proj28

The group meets once or twice a week as time permits and work accomplished
requires.
Mike Rychener is the person who knows the latest news on this subject.

  1.-Meeting times: Usually Mondays and Thursdays, at 3:30 pm.

  2.-Place: Usually 5130 Science Hall

  3. Sending messages to the group

  4. Send a note to Mike Rychener about the next meeting


a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 11. User wants to send note to Rychener. He selects '4'.

```
Type in a message, terminating it with altmode          Proj1      Proj30

                                                    ↑H-BACKSPACE
                                                    ↑U-RETYPE LINE
                                                    $<alt>-DONE
                                                    ↑Q-QUIT

                                                                  ↑C-exit
```

Figure 12. User types note, then altmode, which takes him back to previous frame.

```
Organizational and Procedural issues                    Proj1      Proj20

The group meets once or twice a week as time permits and work accomplished
requires.
Mike Rychener is the person who knows the latest news on this subject.

  1.-Meeting times: Usually Mondays and Thursdays, at 3:30 pm.

  2.-Place: Usually 5130 Science Hall

  3. Sending messages to the group

  4. Send a note to Mike Rychener about the next meeting

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 13. User wants to look at other projects. He types 'r' to get to Proj1.

```
Projects in the Computer Science department                Proj1     Proj1

Research activity in the department is organized and conducted as 'projects'.
Projects may be large, or small, group, or individual - the usual use of the
term denotes a group of people who meet regularly, discuss and work on some
common system or problems.
Current projects are described below under broad subject classifications.

  1. Artificial Intelligence

  2. Programming languages

  3. Operating Systems

  4. Computer system design

  5. Computer system evaluation

  6. Data-base design and evaluation

  7. (Other projects)

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-ZOG ↑C-exit
```

Figure 14. User continues his exploration. When done, he selects ↑C to quit.

This sample interaction gives you the flavor of ZOG from the user point of view. Let us now examine the internal structure of this system by looking at its design and implementation.

4. System Architecture

In order to describe the architecture of the ZOG system, we must distinguish between what a user sees and what a builder of frames sees. It is important to note that any user may become a frame builder; in fact, frame modification and augmentation are the primary means of allowing the user to tailor the system to his needs.

As described in the introduction, the user sees a display of a frame which contains some text and a menu of selections. When the user makes a selection, either by touching the screen or by typing the appropriate character, some action may occur and a new frame may be displayed. The user works his way through a network of frames organized into subsystems, called subnets. A subnet's goal is to guide the user to learn something, by reading the text, or to accomplish some action.

The frame builder sees the same basic system for selection processing, but also sees a communications language that allows him to construct frames and actions to accomplish desired tasks in the frames. The communications language that the frame builder uses has three basic functional capabilities: (1) network positioning to control which frames the user sees, (2) communications control to control interactions between the user and the frames, and (3) network and frame modification to allow existing frames to be modified and new frames to be added.

In Section 4.1, we discuss the design of the mechanisms that support the frame builder's view of ZOG. Since the user's view is a subset of the frame builder's view, we discuss those support mechanisms also. In Section 4.2, we discuss the current implementation of ZOG on the PDP10, which does not support rapid response or a large network. Finally, in Section 4.3, we discuss the adaptation of ZOG to C.mmp, which will support rapid response and a large network.

4.1. Basic System Design

The ZOG system may be viewed as a communications multiplexor. It has a number of logical input devices, each of which may be linked to an arbitrary subset of the logical output devices. The basic function of ZOG is to cycle through the set of input devices and route messages to the appropriate output devices. Any one of these input devices may invoke the communications language, through escape characters, to control the position in the frame network, control the communications multiplexor, or manipulate frames.

The logical input and output devices are listed in Figure 15. The input devices are on the left and the output devices are on the right. Since an input can be routed to any set of outputs, no constraints can be placed on the inputs (i.e., although output devices will make formatting assumptions, ZOG can make no such assumptions). For example, if the input file is routed to selection processing, then the characters being read from the file are treated just as selection characters typed from the keyboard. If

the input file is routed to ZOGNET building, then the file being read is assumed to be in the external frame format (discussed later). If the input file is routed to the subjob, then it can be anything at all. It should also be noted that output devices are not aware of the source of the character streams with which they deal. We will briefly describe each of these logical devices before describing the rest of the system.

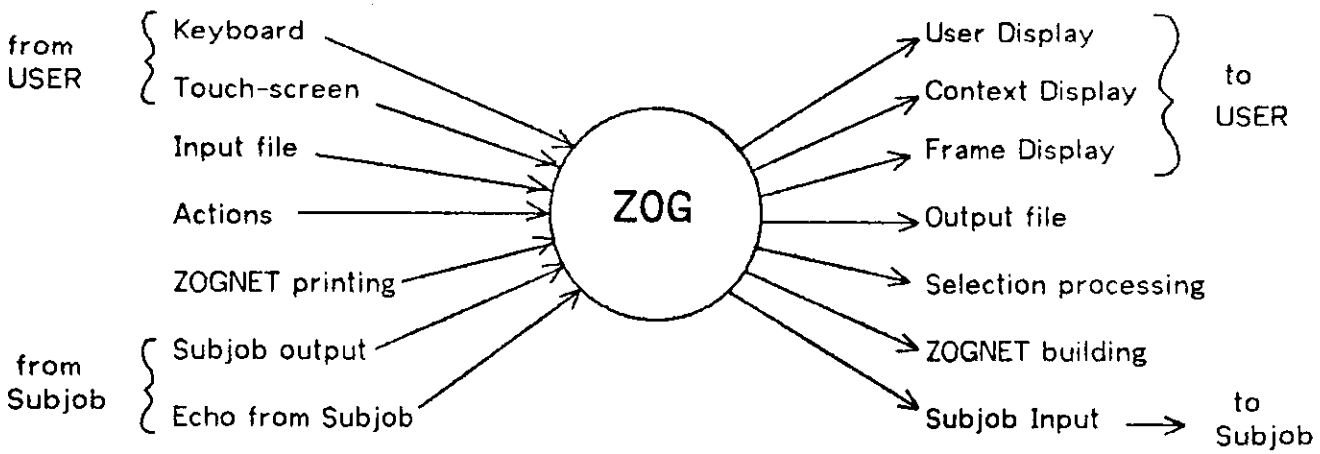

Figure 15. ZOG as a Communication Agent

Input devices generate character streams. The keyboard is read one character at a time. Each character is sent to all output devices linked to the keyboard, unless one of the communications language escape characters is detected. The touch sensitive screen produces a pair of coordinates when touched by a finger. These coordinates are translated into selection characters and then processed as though they had been entered from the keyboard. The communications language escape characters cannot be generated by the touch screen. The action input is processed during selection processing when a selection is made or when a frame is entered. The action is stbred as part of the frame, and its format will be discussed below. The input file (a file as defined and supported by the underlying operating system), once opened, is processed until the end of file is encountered. The subjob is a separate job being controlled by the ZOG job. The subjob may be used to run any arbitrary program, and input from the subjob is terminal output from the subjob's point of view. This job is automatically logged in when the first character is sent to it and logged out when ZOG exits. Subjob echo is produced when characters are sent to the subjob, if the subjob itself is echoing characters. Echo input can not escape to the communications language, even if it contains the escape characters. Finally, ZOGNET printing results

from invoking one of the ZOGNET modification commands in the communications language. The printing is in an external frame format discussed below.

The output devices accept character streams. The selection processing logical output device is what the ZOG user sees most often. It takes a character representing a selection from a menu and does what is appropriate with it (discussed below). Three logical display windows are supported. The frame display is used by selection processing; the user display has no preassigned use (but would normally be used by a subjob); and the context display is a small window used to display a frame number. Each window may include arbitrary areas of the screen. A single output file can be open, and any stream of text characters may be sent to it. Output to the subjob appears as terminal input from the subjob's point of view. Finally, ZOGNET building constructs new frames or modifies existing frames from an external frame format.

When the user first starts interacting with ZOG, there is no file or subjob input, and the keyboard is linked to the selection processor. The system-wide root node, frame one in subnet one (denoted 1.1), is displayed and the user is expected to make some selection.

4.1.1 Frames and Subnets

Let us examine the content and layout of a frame, and the organization of the system into collections of frames, called subnets. From the user's point of view, a frame consists of some text and a menu of selections. The general format he sees has a one line title on the top line, with the frame number in the upper right hand corner. He may also see another frame number just to the left of the frame number; this is the context display. Next, he sees several lines of text. Next, he sees some numbered selections, called options. Finally, he sees some selections along the bottom line, and possibly along the right hand side, which have alphabetic characters instead of digits in front of them. These are called pads. The pads along the bottom are generally global to the whole system, are selected with lower case alphabetics, and are called global pads. The pads along the right side, if any, are local to the frame, are selected with upper case alphabetics (or other characters), and are called local pads.

The internal description for a frame is shown in Figure 16. Each displayed item has positioning information so that the frame builder has maximum flexibility in defining his frame layout. The frame title is specified with its position (vertical and horizontal position on a 80 character by 24 line display) and the text to be displayed. The frame text is specified in the same way. The frame action is a character stream which will be fed to the communications multiplexor when the frame is entered. The option list contains a list of selections whose description will be described below. The local pad list contains a list of selections in the same form. The global pad list contains a list of indices for global pads, which are stored global to the system. The accessor list is a list of frame numbers which reference this frame. It is used primarily for maintenance. The frame identification is the text string which appears in the upper right corner when the frame is displayed. It is constructed by concatenating the subnet name and relative frame number within the subnet. The subnet index and relative frame number

identify the frame. The version number is used for maintenance purposes. Finally, a comment may be stored with the frame. This comment is not displayed when the frame is displayed.

- Frame title: v (line number), h (character position), text
- Frame text: v, h, text
- Frame action: text (not displayed)
- Option list: list of selections
- Local pad list: list of selections
- Global pad list: list of global pad indices
- Accessor list: list of frame numbers (not displayed)
- Frame id: text
- Subnet index: integer (not displayed)
- Relative frame number: integer (not displayed)
- Version number: integer (not displayed)
- Comment: text (not displayed)

Figure 16. Internal Frame Description

The description of a selection, either an option or a pad, is shown in Figure 17. It has position information and the text to display. The displayed text is expected to communicate to the user how to make the selection (e.g., by containing the selector character as the first character of text). The selection may have a next frame, which will be displayed when the selection is made. The selection may have an action, a character stream which is fed to the communications multiplexor when the selection is made. Finally, a selector character is specified. The selection processor uses these selector characters when it tries to evaluate a user selection.

- Selection text: v, h, text
- Next frame: subnet index and relative frame number (not displayed)
- Selection action: text (not displayed)
- Selector: character (not displayed)

Figure 17. Internal Selection Description

Actions are represented as simple text streams. A selection may have an action and a frame may have an action on entry. The text for the action is sent to each logical output device linked to the action input device. Actions may invoke the communications language through appropriate escape characters. Thus, actions may use any ZOG supported facility through the communications language, and may execute arbitrary programs by interacting with the subjob. For example, a set of frames teaching how to use ALGOL could have actions which actually run examples for the user. Because of the open-endedness of actions, they are a powerful means of programming frames.

A set of frames may be organized into a logical network called a subnet. Each subnet has an index (an integer) and may have a print name. The number of subnets allowed is an implementation dependent parameter, but it is expected to be large (hundreds). The number of frames allowed in a subnet is also expected to be large (thousands), but small subnets are not penalized. Under normal circumstances, a user will enter a subnet through its root node (relative frame one). The root node will usually have a frame action which will set the necessary context for the subnet (e.g., start up an appropriate program on the subjob). Also, the root node frame action will normally mark the frame (described in detail later) so that its name appears in the context display and it can be returned to easily.

4.1.2 External Frame Format

An initial design objective was to maintain an external frame format that satisfied a number of desired properties. The format should be readable, allowing someone without a ZOG system to see what is in the frame library. The format should be easily supported by all ZOG systems (different versions and different machines), so that the frame library would be transportable. Finally, the format should be compatible with some existing facility to make its external maintenance and manipulation easy.

The format we have chosen is described in detail in Appendix III. Figure 18 is a sample of the BH format defining the frame shown in Figure 3. It is moderately readable, is easily supported, and is compatible with the BH bibliography system. Although we have reached our objectives with this external frame format, it does fall short of providing total frame library transportability. Frames are transportable if their actions make no use of the subjob. However, if an action evokes some program on the subjob, then that program must be transported as well as the frame. Since those programs are completely arbitrary, we have no control over their transportability. Despite this shortcoming, we feel that this external frame format is reasonable.

```
+A+ 1.1 2
+G+ 1 & 2 & 3 & 4 & 7 & 9
+C+ "System root node"
+T+ T "ZOG - An Interactive Guide System"
+P+ T 1 10
+T+ F "Welcome to ZOG.
You are at the starting place for the entire ZOG network.
ZOG is a guidance system which will help you understand some
subjects and help you to use some programs and systems.

You use ZOG by selecting the options it offers.
Type h (for help) to find out how to use ZOG
or just explore by typing the character in front of any option.
Typing z (for ZOG) will always bring you back here."
+P+ F 3 1
+T+ 01 "1. Table of contents"
+P+ 01 13 3
+F+ 01 1.2
+T+ 02 "2. Utilities for ZOG frame building"
+P+ 02 15 3
+F+ 02 1.11
+T+ 03 "3. Sending a message to the designers of ZOG"
+P+ 03 17 3
+F+ 03 1.13
+T+ 04 "4. Clear frame backup list"
+P+ 04 19 3
+X+ 04 "↑AC↑AD"
+T+ 05 "5. Go to an arbitrary frame - prompts for number"
+P+ 05 21 3
+X+ 05 "↑AG;"
```

Figure 18. Example of BH Format

4.1.3 Selection Processing

Now that we have examined the internal and external structure of frames and subnets, let us take a closer look at the primary mechanism that the user deals with, the selection processor (see Figure 19). Let us start with a displayed frame. The user makes his selection by touching the screen or typing the selector character. In the case of the screen touch, some immediate feedback is given so that the user knows his touch was successful (e.g., a box is drawn around the selection). Then the coordinates are translated into a selector character which is treated as though it were typed. The selection processor then scans options, local pads, and finally global pads trying to match the selector character. If no match is found, it rings a bell. If a match is found, the selection is evaluated.

1. Get selector character.
2. Find selection. If none, ring bell and back to 1.
3. Interpret selection action, if any.
4. Set next frame, if any.
5. Display frame, if changed.
6. Interpret frame action, if any and if changed.
7. Back to 1.

Figure 19. Selection Processor Cycle

To evaluate a selection, the selection processor first checks if the selection has an action. If it does, then the character stream from the action is sent to each logical output device linked to the action input device. The selection processor handles escape characters to the communications language as well. After the action has been interpreted, the selection processor checks if the selection has specified a next frame. If so, it saves the current frame on the frame backup list and sets the specified next frame as current. Finally, the selection processor checks if the current frame is different from the last frame. If it is different, it is displayed and checked for any frame action. If the frame has an entry action, it is interpreted.

4.1.4 Communications Language

The communications language allows the user and the frame builder to maintain control over the interactions between the user and the various parts of the ZOG system. It can be invoked with escape characters from the keyboard, input file, subjob, or actions. It provides three basic facilities: (1) network positioning (escape character ↑A (control-A)); (2) communications control (escape character ↑B); and (3) frame modification (escape character ↑D). This language is very simple: it has no variables, no conditionals, and no repeats. It is invoked by one of the three escape characters mentioned. The character following the escape character is a command. Some commands take operands by examining the character stream following the command. Appendix I describes the language in detail.

The network positioning commands (↑A) provide support for moving through the network in ways other than simple selection. Whenever a new frame is selected, the previous frame is saved on a backup list. The positioning commands allow the user to back up to the previous frame, mark a frame in the backup list, return to the last marked frame, and clear the backup list. There is also a composite command which backs up to the previous frame and takes the next option. Finally, there are commands to re-display the current frame and go to an arbitrary frame.

The communications control commands (↑B) provide support for maintaining and modifying the routing control information for the logical input devices. These commands also support opening and closing the input and output files, manipulating the subjob, exiting from or saving a ZOG system, and manipulating the display in a terminal independent way.

The frame modification commands (↑D) provide a basis for an editing system. There are commands for sending sets of frames, single frames, or parts of frames. These may be used to save frames externally (e.g., if the ZOGNET printing input is routed to the output file) or to send the frames to a more elaborate editor running on the subjob. Frames may be constructed by routing the appropriate input device (usually the input file or subjob) to the ZOGNET builder output device. There are commands for saving and restoring frames, to allow for potentially temporary changes to frames. There are commands to display all or part of an arbitrary frame. There are commands to delete all or part of an arbitrary frame. There is a command to move a frame from one part of the network to another. There is a command to enter a frame creation mode that prompts for the most common parts of a new frame. Finally, there is a command to enter an interactive alter mode for a frame. The alter mode and create mode, combined with a set of frames which aid the user in using the other frame modification commands, form the frame editor, called ZED.

4.1.5 Frame Editing -- ZED

The frame editor, called ZED (for ZOG Editor), is composed of three major parts: (1) a set of frames that allow you to delete frames, move frames from one subnet to another, and create new subnets; (2) a creation mode which guides you while building new frames; and (3) an alter mode for altering a single frame. Appendix II has complete specifications for ZED.

The set of frames simply provides a convenient interface between the user and the communications language frame modification commands. Although the frames are not strictly part of the basic ZOG system, it is useful to think of ZED (frames, create mode, alter mode) as a single system.

The create mode is used in conjunction with a special frame, presented when the user accesses a frame not yet defined, to make top-down subnet construction easy. On access to a non-existent frame, a frame is displayed which tells the user he has reached a non-existent frame, and provides him with two pads: back to back up and create to enter the create mode. In create mode, the user is prompted for the most

common parts of a new frame, and is shown the target frame as it is being constructed. Once a frame is constructed, the user may select one of that frame's options which may point to a new undefined frame. By systematically stepping through the options and creating new frames at each step, subnets can be created in a top-down fashion.

The alter mode is accessed through the alter global pad which appears on each frame. The design for alter mode was derived from extending the alter mode of SOS (a line oriented text editor found on many PDP10 systems (Weiher and Savitzky, 1970)) from one dimensional line editing to two dimensional frame editing. The frame being edited is always displayed. Alter mode commands are typed (but not echoed), and the results of an editing operation appear as changes to the displayed frame.

4.1.6 User Profiles

A user profile, in general, is a user-defined body of information that specifies how that particular user will use a particular computing system. The purpose is to allow the user to tailor the system to meet his own needs or style. It is most commonly implemented as a script that is executed when the user first enters the system. In ZOG, it is implemented by allowing users to modify any frame in the system (with ZED) so that when they enter ZOG, their personal versions of modified frames will be used.

For example, when a user makes a lot of use of one particular path through the network, he might like to shorten that path to make the system easier and faster for him to use. To make this possible, we allow the user to edit any frame. When he does so, his edited version is maintained in a user profile file which the system reads when it is started. When he next uses the system, his version of a particular frame will override the system version, unless the system version has since changed. If it has changed, the user will be notified. This mechanism allows the user to adapt the frame library to his personal needs in a natural and flexible way.

It should be noted that the PROMIS system does not have this feature, and they have received some criticism from their users because of it. The ultimate value of such a feature is not clear in this kind of communications system. It probably depends a great deal on the task domain. We include user profiling in ZOG to allow us to evaluate its desirability in a number of different task domains.

4.1.7 Conventions

We have adopted a number of conventions which make the system easier to use for both the user and the frame builder. They deal with the frame display organization, selection format, and commonly used pads. The system does not enforce these conventions in any way.

The organization of the displayed frame has only two system enforced decisions.

First, the frame identification is always displayed in the upper right corner. Second, the context display is always just to the left of the frame identification. The context display shows the frame identification of the last marked frame. The layout of the title, text, options, and pads is completely up to the frame builder.

Our current display format is shown in Figure 20. It assumes a display with 24 lines of 80 characters. The title is restricted to 56 characters on the first line. A blank line separates the title and text; the text is one to nineteen lines long with 80 characters per line. The number of options is limited to nine so that digits may be used for selectors. Options share space with the text. That is, the less text you have, the more options you may have. Each option is preceded by a blank line. The options may be up to 56 characters long. Space on the right is reserved for local pads, which may be 13 characters long. The most common pads (global pads) are on the bottom line. Options are selected by digits, local pads by upper case alphabetics, and global pads by lower case alphabetics. Any selection which is inert (has no action or next frame) should be prefixed with a minus sign to indicate that it is not worth selecting. Finally, if a frame needs more than nine options, it should be broken into several frames with the ninth option of each pointing to a frame with the continuation of the list.

```
[TITLE.................................56.....]  [MarkID.11.]  [FrameID.11]

[TEXT.....................................................................]
|      Text-lines + 2*Number-of-options + Workspace-lines = 19             |
[.....................................................................]

 1. [OPTION-TITLE.............................56.....]    A-[PAD-TITLE.13]
(1. PRIOR options exist          0. START of options)
 2. [OPTION-TITLE.............................56.....]    B-[PAD-TITLE.13]

 3.-[OPTION-TITLE.............................56.....]    C-[PAD-TITLE.13]

 . ..................................................  .. ............

 8. [OPTION-TITLE.............................56.....]    H-[PAD-TITLE.13]

 9. [OPTION-TITLE.............................56.....]    I-[PAD-TITLE.13]
(9. MORE options exist)
[WORKSPACE FOR SUBJOB OR LOCAL NET.....................................]
|                                                                      |
[.....................................................................]

a-alter  b-back  d-display  h-help  m-mark  n-next  r-return  z-zog ↑C-exit
```

Figure 20. Conventions for Display Layout

Another unenforced set of conventions has to do with the global pads. In Figure 20, a set of global pads is listed on the bottom line. The alter pad enters the alter mode of the frame editor, ZED. The back pad returns to the previous frame. The display pad re-displays the current frame. This is useful if the display has been disturbed by some external cause (e.g., a terminal failure). The help pad enters a subnet which attempts to describe how to use ZOG. The mark pad marks the current frame. The current frame identification will appear in the context display and the next selection of the return pad will bring the user back to this frame. The next pad backs up to the previous frame and automaticaly selects the next option. If there is no next option, then it behaves just as back does. The return pad returns to the last marked frame, whose frame identification is displayed in the context display. The ZOG pad goes to the root node of the whole system without disturbing your context (i.e., you can get back). The exit pad forces an exit from ZOG and logs out the subjob if it was logged in. Note that the exit is done by ↑C, which is the standard monitor escape command on the PDP10 (a monitor CONTINUE command will put you back in ZOG where you left).

The root node of a subnet has some special properties by convention. Its frame action automaticaly marks the root node for easy return and to supply the user with context information. It normally does not have mark and return pads. It may or may not have the back pad. If the back pad is missing, there will be update and quit pads. The quit pad will abort whatever action the subnet is attempting and do the equivalent of back. The update pad will complete the subnet action and then do the equivalent of back.

4.2. Current Implementation

The current implementation of ZOG is written in L*(I) and runs on a PDP10. L*(I) is an interactive system-building system (Newell, Freeman, McCracken, and Robertson, 1970) that has made implementation and experimentation with ZOG very straightforward. The current implementation follows the design described in the previous section with only three exceptions: (1) user profiling is not currently implemented, (2) rapid response has not yet been achieved, and (3) the touch screen input and graphics output are not implemented.

The current implementation does support a number of display terminals, including three kinds of Beehives (SuperBee, MiniBee, and B100) and the Tektronix 4023. It also supports a mode which may be used on any other kind of terminal (including a teletype), but with the display format only approximated. Support for other display terminals may be easily added, assuming the terminal in question has cursor addressing, screen clear, and backspace functions, and has at least 24 lines of 80 characters.

The next section will describe a design for the next version of ZOG, which achieves the rapid response goals by utilizing a specialized terminal and moving the system to a different computer. We plan to maintain the PDP10 version of ZOG so that

users who do not have access to the specialized ZOG terminals will be able to continue using the system.

4.3. Rapid-response Large-network Design

Our goal is to provide rapid response with large networks on an existing time-sharing system with access to the ARPA network (Heart, Kahn, Ornstein, Crowther, and Walden, 1970). Because of the nature of time-sharing, it is not possible to achieve guaranteed response levels without getting into the lowest levels of the operating system and modifying it. This option is not available to us with our PDP10 operating system. However, we do have that option on another computer in the CMU environment, C.mmp. In this section, we discuss the design of such a rapid response version of ZOG for C.mmp.

C.mmp is a multi-mini-processor (Wulf and Bell, 1972) being developed at CMU. It has sixteen PDP11's connected through a crosspoint switch to a large shared memory. The operating system for C.mmp is called Hydra (Wulf et al, 1974). Because of the experimental nature of Hydra, it is possible for us to modify the lowest levels of it to help achieve our response goals and still live within a time-shared environment with access to the ARPA network.

Transfer of ZOG from the PDP10 to C.mmp will be straightforward since ZOG is written in L* and there are essentially compatible versions of L* on the PDP10 and C.mmp. The major work in transferring the system will deal with adapting to the specialized ZOG terminal, and modifying Hydra and ZOG to cooperate in achieving the rapid response goals.

4.3.1 ZOG Terminal

The key to rapid response is the terminal being used. Conventional terminals are limited to relatively low bandwidths between processor and terminal. To get the kind of response we desire, it is necessary to use terminals that permit transfer rates of 50 kilobaud or higher (see below for derivation of this requirement). Our choice for this terminal is a graphics system developed at CMU, called the GDP (Graphics Display Processor, (Rosen, 1974)). The GDP is a vector drawing graphics system which can change the entire screen in less than 16 milliseconds. To allow full utilization of this speed, the GDP is being interfaced directly to one of the PDP11's on C.mmp, with software support for it built into Hydra.

A ZOG terminal is a combination of a C.mmp GDP, a touch screen, and a high speed disk for local frame storage. The touch screen is a clear plate of glass placed over the face of the GDP. When touched by a human finger, it produces a pair of coordinates with a 1/10 inch accuracy. Low level software support for the touch screen must also be added to Hydra.

Finally, a ZOG terminal has a high speed disk for frame storage. The disk being used is an IMS disk (.5 megaword capacity), and has a special feature that allows rapid access to pages (4,096 16-bit words). The feature allows page transfers to begin with zero latency. This feature will aid in reaching the rapid response goal while allowing a large frame network to be stored on secondary storage.

4.3.2 Rapid Response

Rapid response will be achieved by augmenting the interrupt service for the touch screen (part of Hydra) to handle most of selection processing. When a touch is made, an immediate response is given by drawing a box around the selection. This gives the user immediate feedback to confirm that his touch worked. The interrupt driven selection processor would then find the selection and evaluate it. If a selection action or frame action is encountered during evaluation, then the basic ZOG system would be used an non-interrupt level. However, for a selection with no action, the next frame would be found and displayed at interrupt level. This design avoids most of the overheads normally imposed by an operating system.

Interpretation of actions will be handled as in the current implementation. This implies that a network which has a high percentage of actions will suffer and will not enjoy rapid response. If this becomes too severe a problem, the action interpreter could also be moved to interrupt level. However, modifications to these interrupt drivers are much more difficult to make than modifications to a user level ZOG system. Because of this, we will avoid moving action interpretation to interrupt level as long as possible.

It is useful to compare our rapid response requirements and techniques with those of PROMIS. In PROMIS, the desire is to support about 30 terminals with a dedicated system so that each user experiences .25 second response 70% of the time. The average frame size for PROMIS is 800 10-bit characters. The 250 milliseconds is broken down to 125 milliseconds for disk accesses, 50 milliseconds for processing time (to find frame and construct display), 50 milliseconds to transmit the display to the terminal, and 25 milliseconds to spare. This translates to a requirement of 300 kilobaud peak transfer rate between processor and terminal for the PROMIS system.

In ZOG, the desire is to support one terminal per processor (because of the nature of the Graphics Display Processor) in a time-shared system so that the user experiences .05 second response 70% of the time. The average frame size for ZOG is 700 8-bit characters. The 50 milliseconds is broken down to 16 milliseconds for a cache access (from the IMS disk, discussed below), no time for processing, 16 milliseconds to transmit the display to the GDP terminal, and 18 milliseconds to spare. This translates to a requirement of 50 kilobaud transfer rate between processor and GDP, which is easily met. Note that ZOG frames are pre-processed to eliminate the processing step during the selection cycle. This design decision was made to allow us to explore the rapid response dimension more freely. It may limit the quality of what is displayed in ways that will eventually force us to use the same kind of frame processing that PROMIS currently uses. The single terminal per processor target for

ZOG was derived from equipment and techniques that we have access to, and that allow us to fully explore the response time issues. The issues of how to extend to more terminals and more cost-effective equipment will be ignored until we understand the more basic issues.

4.3.3 Large Network

In order to support a large frame network, we have designed a three-level memory hierarchy. At the lowest level is the primary memory to which the GDP has access. At the second level is the IMS disk which will be used as a cache for frames. At the highest level is bulk secondary storage.

Each subnet has a set of tables which indicates where each frame in that subnet resides (which of the three memory levels). The interrupt driven selection processor will use these tables to find the frame. If the frame is in primary memory, then the selection processor simply needs to change the GDP display. This takes less than 16 milliseconds. If the frame is in the cache (IMS disk), then a transfer is initiated. A full page transfer takes approximately 16 milliseconds. When the transfer is complete, the IMS disk interrupt service will return control to the selection processor which will then change the GDP display. The total time in this case is less than 32 milliseconds. Finally, if the frame is on secondary storage, a request will be made to transfer the page it is in to the cache. This will be followed by a transfer to primary memory and the GDP display. This case may require hundreds of milliseconds.

Assuming an average frame has 700 characters, about six frames may be stored in a page in the cache. The cache capacity is 128 pages, or about 768 frames. The total frame library is expected to be as much as 50 times larger than the cache (40,000 frames). Thus, it will be important to try to organize the pages to maximize the cache hit rate. It may also be possible to do some look ahead in transfering between secondary storage and the cache. At the moment, we have no data on these operations, and the success of this memory hierarchy is still an open question.

We have described the basic ideas of ZOG, given an extended example of its use, and examined its design and implementation. Let us now look at the potential and cost of this communication philosophy.

5. The Potential of ZOG

The most important question is why ZOG (nee PROMIS) has the potential to be a new communication interface, especially since menu selection techniques are common practice. The answer lies in rapid response into a large network (supported by the other principles). It produces a man-computer interface with qualitatively different properties, best summed up by the slogan of "transparency with speed".

The basic advantage of menu selection is that the user does not have to bring to the situation knowledge either of the functional possibilities for interaction or how to communicate them. Either of these can be a strong barrier to communication. Menu selection instructs while operating. It is superior to a manual in two ways. (1) It eliminates the jump from barely acquired knowledge to how to put it into action. With a manual the user must still decide exactly what to do after reading the textual material; in menu selection he simply does it. (2) It eliminates the search for the relevant information by locating the relevant knowledge at the site of action. Manuals are not the only alternatives (e.g., there are instruction sheets, scripts and on-line help facilities); our purpose is only to make clear the essential dimensions of menu selection.

Menu selection has two disadvantages as normally implemented. First, it is slow. Expressed as channel capacity, in menu selection the user can select about 1 out of 10 alternatives every 5 seconds (i.e., about 1 bit per second). In typing, by contrast, the rate is about 1 word per second for a reasonable typist (i.e., about 10 bits per second). These figures are exceedingly rough, since they depend on many factors. However, they serve to illustrate that one can get messages into a computer about an order of magnitude faster by typing than by menu selection. The second disadvantage is the forced interposition of explanatory text and options. Menu selection becomes especially trying for skilled operators who no longer need the instruction that it offers. A manual can be put away after it is learned. This disadvantage compounds the speed disadvantage, since, even though the text can be ignored, there are usually extra options to wade through. However, the negative reactions probably are generated by more than just slowness, but also by a sense of needless bother and frustration about being forced to operate inefficiently.

These disadvantages do not prevent menu selection from being a useful technique. They do conspire to limit its use to applications where the user is a novice so that he needs explanation and could not communicate rapidly in any event. The relevant domain is broader than might be thought, ranging from rare situations, where everyone is a novice, to systems used repetitively by experts who do not wish to acquire technical jargon.

Rapid response and the use of large networks are designed to remove both of these disadvantages. A response time of .5 seconds, as opposed to 5 seconds, gives back the missing factor of 10 in speed. Whether communication speed can be then competitive with typing is not determined simply by such raw figures, of course. But now, at least, the issue is a matter of details and not a foregone conclusion. Permitting avoidance of unneeded explanatory steps is basically a question of designing

alternative sequences, so there are "short circuits" for experts and "long circuits" for novices. There are two enabling conditions for this. One of these is the large network. Providing many alternative paths requires many frames. The other condition is the ability to create frames and design networks easily.

Grant for the moment the major technical premise, that these two disadvantages can be effectively nullified. Then rapid selection, large network menu selection systems become a distinct type of man-machine interface. They maintain their properties of ease of use and transparency, but now they permit both expert and novice operation, and in whatever the user's mixture of experience and familiarity with different aspects of the system. They become a general purpose interface.

To date the entire evidence for the claim of a distinct type of interface comes from the experience of the PROMIS system. This is substantial, however, and a brief recounting of that experience is useful at this point.

PROMIS has grown out of a long standing effort of its leader, Dr. L. Weed, to develop and promote a problem oriented approach to health care delivery (Hurst and Walker, 1972). The basic tenet of this approach is that the patient record should be organized according to patient problems rather than the traditional organization by data source. Using this different organization not only reduces wasted time in health care delivery, but also provides improvement in care because the doctors, nurses, and technicians all become aware of the patients' problems and what is being done to cope with them. This problem oriented approach has been in use in many places for many years, quite independent of any attempt at computerization.

Development of a computer system (PROMIS) as the underlying technology for the problem oriented medical record was initiated about six years ago by Dr. Weed at the University of Vermont with support from the Dept. of Health, Education, and Welfare. The group developed an integrated system approach that includes as its central core a rapid response large network menu selection technique, such as we have described. The design is responsive to the demands of establishing an automated system as the sole informational vehicle for hospital medical practice. These same demands are faced in many computer applications, but are not any less important for that. A primary concern is the range and diversity of capabilities, sophistication, motivation and authority of the users. PROMIS' solution of avoiding typing, providing complete access via self-explaining menu selection, and getting high enough speed for expert use is radical and required the supporting notions of rapid response, touch screens and large networks to make it feasible.

A first version of the PROMIS system was designed and implemented using a CDC 1700 and a specially designed CDC terminal with a fixed set of touch pads. This system became operational in 1972 on a gynecology ward, providing the complete patient record. About three years of continuous experience was obtained with this system. A second completely reworked hardware-software system is at an advanced stage of development, using a network of Varian V75s. The critical specifications for the user interface -- rapid response, touch screen, large network -- have all remained. The network of medical knowledge has been converted. The new system is primarily responsive to the evolving computer technology, and the needs for efficiency, cost-

effectiveness, reliability, portability, etc. It should be reemphasized that PROMIS is an integrated system for doing the information tasks of patient care. The user interface, though perhaps its most striking novelty from a computer system viewpoint (and the focus of our interest), is only one aspect of the total system.

The experience with PROMIS is highly supportive of the claim made above. However, no detailed analysis of the experience with the interface exists, owing primarily to the subordination of all aspects of PROMIS to the overriding goal of developing the total system. This is unfortunate, for we are thrown back mostly on personal and anecdotal experience. (We note in passing that other interface schemes are hardly if at all better analyzed in the literature.) It can be said with some surety that the PROMIS interface works well as a primary interface at both the novice and expert level, that the frame library provides a world of medical knowledge that is largely sufficient to the total problem of medical care on the ward, and that the interface strikes one as highly novel and exciting. We do not need definitive evidence from PROMIS, of course, only enough evidence to support the serious exploration into the interface scheme.

In assessing the evidence from PROMIS it is legitimate to wonder why, if the interface is so effective, it has not been widely copied and adapted. The answer is composed of the combination of several factors: the technical demands and expense of the system; the expense and novelty of developing large nets; the embedding in a larger total system, making extraction difficult; and the location of PROMIS in the medical world, outside of the computer main stream. None of these affect the intrinsic merits of the communication philosophy. However, the two cost factors are important for our purposes, since they also affect ZOG. They will be taken up below in a general discussion of costs.

The claim just made for ZOG is a minimal one. We will also lay out a much stronger claim, which might be taken as a maximal claim. In doing so, our intent is to describe the potentialities of ZOG. Though convinced of the potential of ZOG and desirous of exploiting it, we are strongly motivated by the desire to understand the interface scientifically, to discover its limits as well as its strengths. The maximal claim is that the ZOG type of interface is a preferred mode of man-machine interaction, even over the use of natural language dialog with the computer in the role of intelligent agent. To understand this claim, we must step back a little.

There are two polar views about how to structure man-machine interaction. One is the computer as tool. Under this view one wishes to make the computer a better tool -- more responsive, easier to wield, more reliable in application, capable of doing a bigger job at a stroke. Control remains with the user. The other view is the computer as intelligent assistant. In this view one wishes to make the computer more intelligent and communication with it more natural. One does not wield an intelligent assistant, one tells it what one wants. Intelligent agents figure out what is necessary and do it themselves, without bothering you about it. They tell you the results and explain to you what you need to know.

There is a tension between these two views, for in an important sense intelligence is obscure. More precisely, intelligence in oneself is illuminating and

transparent. In others, it is obscure and inpenetrable. This follows from the tautalogical principle that to understand another's act of intelligence requires an act of intelligence. And precisely what an intelligent assistant is supposed to provide is freedom (of the user) from the effort of understanding. Put one other way, delegation requires an act of faith.

This trade-off-like opposition between computer as tool and computer as intelligent agent is a fairly deep affair, certainly capable of sustaining more analysis than we can devote to it here. We wish to use it only to make clear a claim about a ZOG-like interface. We make no assertions whether the tension is unresolvable or whether the computer might not permit combining these two views in many as yet unforeseen ways.

ZOG is an evolution in the tool direction. It seeks to produce a transparent device which, in itself, has no intelligence at all, but is immensely responsive to the user. It seeks to do this in the arena where we normally expect to use natural language, namely, dealing with large bodies of knowledge. Indeed ZOG uses natural language for its output (though arranged in a sort of spatial dialogue), for the user has good devices for assimilating it. However, its own internal structure, which governs what it says and when it choses to say it, is completely open to examination by the user. In fact, examination takes place as a simple side effect of requesting the knowledge from the user.

The maximal claim then is that this mode of communication will be substantially superior to that of communication with an intelligent agent operating within the frame of the usual natural language dialog. The control exercised by the user and the ability to acquire the relevant pieces at a rate matching the users capabilities (no matter how fast) will more than offset the communication capabilities provided by intelligence applied to real time dialog.

This claim does not assert the unimportance of intelligence. A good network results only from intelligent analysis of the topic -- it is frozen intelligence. The claim refers to the process of communication, to the efficiency of knowledge acquisition or complex process control by a human user. Nor does this claim make an assertion about the scope of its application. Dynamic situations surely exist in which time to construct a network must be thrown in the balance against time to communicate by some more direct way from the naturally occurring situation.

This maximal claim, if sustained, could fairly be called revolutionary. What grounds are there for thinking the interface might contain the seeds of such an eventuality, rather than simply the minimal claim of being another (different and useful) variant in the armatorium of all interfaces for computers? The unique property of ZOG-like interfaces is the rate of change of visual textual (and pictorial) information under user control. We can think of no other situation where this particular high degree of informational selectivity is evident. We can expect it to yield some quite new communicative phenomena. Other dynamically controlled visual situations (as in Sketchpad (Sutherland, 1963)) operate at a lower interactive rate informationally (and are unique in other ways). The computer, as its speed and memory increases, opens a continual sequence of genuinely new interactive experiences. The rapid response large network interface is simply one of these.

6. Cost

We have discussed the functional potential of ZOG. It is also necessary to discuss the costs. ZOG (and PROMIS) are expensive systems in two ways: the cost of the hardware, and the cost of preparing the networks. The appropriate form to discuss costs is in terms of system demands for processing and memory, and manpower demands for networks. Reduction of these to dollars confounds the issue with particular technology and minor design decisions.

The technical demand of rapid response ranges from 50 to 300 kilobaud peak transmission, depending on implementation techniques. The unpredictability implied by the large network essentially means this rate must be available from large storage devices, though of course it can be shared for all terminals. Such a data rate is by no means out of the question, but it is like disk-core transmission rates and is very high for terminals. The technical demand for the large networks is about 25 megabytes of storage (700 characters per frame times 35,000 frames). Again, this is not out of the question, but is substantial. However it is mostly shared among all terminals. Touch screen capability is itself not common on terminals. The CDC terminals were very expensive, reflecting an earlier technology; the current technology (from Instronics, Inc.) which is wedded to a regular alphanumeric terminal is still moderately expensive. But here demand could no doubt bring the cost under control. In summary, compared to currently popular interfaces the PROMIS interface looks unattractive; only substantial conviction of its useful properties and/or a strong change in technological costs would lead to its exploitation.

The cost of developing the large network may be by far the largest barrier to the adoption of this philosophy. Under current art (that of PROMIS) each frame must be hand crafted by a professional skilled in the knowledge embedded in the frames. The extra skill to be a good frame and net designer is essentially unknown, for the requirements of this skill are quite unknown. Certainly it is much less than content skill (i.e., more easily learned). The PROMIS frame library was built as a rate of about two frames per man-day of professional time. The 35,000 frame library required about 70 man-years to generate. There are additional issues of quality control, debugging, and testing. They may double the time. PROMIS estimates it has 100 professional man years invested in its frame library. This tremendous cost factor implies that work done on machine aided (or semi-automatic) frame generation will become critical to the adoption of this philosophy.

7. Next Steps

From the general description we have given, the main lines of study and development needed to understand rapid response large network communication interfaces are apparent:

(1) Construction of large networks for several domains with varying task characteristics.

(2) Development of techniques for constructing large networks.

(3) Exploration of the human performance as a function of design parameters, to enable optimization of the design and evaluation of trade-offs against engineering costs.

(4) Evaluation of the total system performance against alternative schemes for accomplishing the same communication goals.

(5) Design studies to explore system costs and efficient implementations.

This is not the place to lay out detailed plans. However, a few general remarks are appropriate.

Different task domains impose different patterns of demands. Our initial application domain is as a guide system to the facilities of the Computer Science Department at CMU, including the program facilities on our computers. This combines top-down exposition plus guided use of programs plus cross indexing. Possibly it will exercise user profiling, though that is not certain yet. Another potential application is as a command language for C.mmp. This would put more stress on ZOG's ability to be used routinely in an operational mode. Neither of these involves the growth of the frame network as an essential feature. The use of a ZOGNET to represent a design specification or a project plan where the net is to be built by the participants themselves, would lay stress on the net-construction activities. The point is only that a range of applications will be required in order to uncover the ways a ZOG interface can be used and understand its strengths and weaknesses.

We noted the large cost of developing the nets. Discovering aids to this process is essential. Nets themselves appear to have much redundant structure, stemming from the advantage to the user of predictability in how to use the net. With the development of some principles of net construction is seems possible to build some aids to construction. How much of the burden they might lift is hard to know at this point, especially when the knowledge resides in the net-builder's head, so must be entered into the machine by a linear typing processing in any event. Organizing the net is a significant aspect of net building and here construction aids might be quite useful. This latter could also assure aspects of net quality by consistent adherence to certain organizational schemes.

The next two items jointly show the need to understand the performance of the total system consisting of a user+ZOG+ZOGNET. The third item reflects the internal concern with discovering the best way to construct a ZOG system (e.g., how important is the frame response time, how should the display be arranged). The fourth item reflects the performance of the total system. Though some human factors work has been done on man-machine interfaces, the current art does not offer good theories or data for these tasks (it does provide good experimental methodology for answering specific questions). Our approach is to try to construct an information processing model of the user's behavior (Newell, 1977) that would give us some leverage on both these issues.

The final item involves exploring the implementation space. The high system costs assures that this is a problem. However, it is probably the least of all the problems on the list. The ultimate appeal of ZOG-like schemes is in the quality of interaction they offer, so that establishing the performance aspects and developing an appreciation for them will be the priority issues.

Appendix I. ZOG Communications Language

Under normal circumstances, ZOG routes character streams from a number of sources of input to a number of logical output devices. Any one of these input sources may invoke the communications language with one of three escape characters: ↑A (control-A), ↑B, or ↑D. These escape characters provide support for the three basic facilities that ZOG provides: ZOGNET positioning, communications control, and ZOGNET modification. After the command has been executed, the character stream continues to flow as determined by the new state. The following is a complete summary of features of the communications language (with the appropriate escape character shown before each command).

I.1. ZOGNET Positioning Commands

↑A↑A - Send ↑A as though no escape occurred.

↑AD - Display current frame. This command is useful if the display area becomes cluttered and the user wishes to redisplay the frame.

↑AM - Mark current frame for later return.

↑AR - Return to the last marked frame.

↑AB - Back up one frame.

↑AN - Next option. This command goes to the previous frame and selects the option following the one last selected. It is useful while exploring an unfamiliar network.

↑AGm.n; - Go to frame n in subnet m.

↑AC - Clear backup list. Each time the user makes a selection, the frame he was at is saved in a backup list. This list is used by Backup, Mark, Next, and Return to preserve state. The Clear command empties this list, with the side effect of removing any "marks".

I.2. Communications Control Commands

↑B↑B - Send ↑B as though no escape occurred.

↑BP<input> - Push input route. The current list of outputs to which the specified input are routed is saved. <input> is a single character: F for file, K for keyboard, T for touch-screen, A for action, Z for ZOGNET printing, J for subjob, and E for echo from subjob.

↑BU<input> - Pop input route. Restores the list of outputs for the specified input, assuming that a ↑BP<input> was previously done for that input. Otherwise, it does nothing.

↑BR<input><output1>...<outputn>; - Route input to outputs. This command establishes a list of outputs for a specified input. <outputm> is a single character: S for selection processing, Z for ZOGNET building, J for subjob, F for file, U for user display, D for frame display, and C for context display (are for "marked" frame number).

↑BI<filenam.ext>; - Open input file. Once opened, an input file will be read to its end, with each character being sent to each output devices specified in the file input route. ↑BI; will cause a prompt to be printed and the file name to be obtained from the keyboard.

↑BO<filenam.ext>; - Open output file. Any output directed to the output file before it is opened will be ignored. ↑BO; will cause a prompt to be printed and the file name to be obtained from the keyboard. If the file already exists, the user will be given a chance to abort the open.

↑BC - Close output file.

↑BM - Place subjob in monitor mode. This is equivalent to sending a series of ↑C's to the subjob.

↑BE - Exit from ZOG. If a subjob has been logged in, this will logout the subjob. If an output file has been opened, this will close it. ↑C is trapped by ZOG and will cause a ↑BE unless the keyboard is routed to the subjob, in which case it will be passed on to the subjob.

↑BS<filenam.ext>; - Save ZOG. This command will save your current core image on specified file. ↑BS; will prompt for file name.

↑B? - Print routing information. This command will print a map of where each input is currently routed. It also prints information about opened files.

↑BDv,h; - Position display cursor. This command takes a vertical line number, v, from 1 to 25, and a horizontal character position, h, from 1 to 80, and moves the cursor to that location. When ZOG is started, it asks the user what kind of terminal he is using. This allows actions and subjobs to be terminal independent.

↑B. - Clear display.

I.3. ZOGNET Modification Commands

↑D↑D - Send ↑D as though no escape occurred.

↑DFm.n; - Send frame n in subnet m. This command sends the external (BH) form of the specified frame to outputs linked to the ZOGNET printing input. The BH form is terminated by a ↑Z.

↑DFm.n,<desc> - Send part of frame n in subnet m. This command behaves as ↑DFm.n; except that it sends only part of the frame. <desc> is: T for frame title, F for frame text, X for frame action, C for frame comment, or S<char> for a selection (option or pad).

↑DGn; - Send global pad n. This command sends the BH form of the specified global pad.

↑DSm; - Send subnet m. This command sends the BH form of all frames in the specified subnet.

↑DX - Send all global pads. This command sends the BH form for all global pads.

↑DT - Send all subnet descriptors. This command sends BH descriptors for all subnets. A subnet descriptor specifies the subnet index and print name.

↑DW - Send whole ZOGNET. This command sends all subnet descriptors, all global pads, and all frames in BH form.

↑DAm.n; - Send accessors to frame n in subnet m. This command sends the BH form for each frame which accesses the specified frame.

↑DC - Send current frame number. This command sends "m.n;", where the current frame is number n in subnet m.

↑DNm; - Send new frame number for subnet m. This command sends "m.n;" where n is the next free frame number in subnet m. It sends "0;" if no room is left.

↑DZ - Send new global pad index. This command sends "n;" or "0;" to specify the next free global pad index.

↑DUm; - Send name of subnet m. This command sends "name;".

↑DVname; - Send index of subnet with specified name. This command sends "m;".

↑D?m.n; - Does frame n in subnet m exist? This command sends either "Yes;" or "No;".

↑DPm.n; - Preserve frame n in subnet m. A copy of the specified frame is made and placed on a central list of preserved frames. The version number of the frame is updated.

↑DR - Restore latest preserved frame.

↑DC - Cancel latest preserved frame. This command will erase the preserved frame. Also note that ↑AC will cancel all preserved frames.

↑DYm.n; - Display frame n in subnet m. This command displays the specified frame in the same way the selection processor displays frames.

↑DYm.n,<desc> - Display part of frame n in subnet m. This command displays the specified part in BH form. <desc> is the same as for ↑DFm.n,<desc>.

↑DDm.n; - Delete frame n in subnet m.

↑DDm.n,<desc> - Delete part of frame n in subnet m. <desc> is the same as for ↑DFm.n,<desc>.

↑DMm.n,m.n; - Move first frame to second location. This commands is a destructive move; i.e., the source will no longer exist, any frame residing at the destination before the move will no longer exist, and the moved frame will have its frame name changed to fit its new location.

↑DIm.n; - Enter ZED create mode for frame n in subnet m.

↑DEm.n; - Enter ZED alter mode for frame n in subnet m.

Appendix II. ZED - The Frame Editor

The frame editor, called ZED, is composed of three major parts: (1) a set of frames that allow you to delete frames, move frames from one subnet to another, and create new subnets; (2) a creation mode which guides you while building new frames; and (3) an alter mode for altering a single frame.

II.1. ZED Frames

The root node of the ZED network provides the following options.

1. Alter any frame. If selected, this option prompts for a frame number and enters alter mode.

2. Create any frame. If selected, this option prompts for a frame number and enters create mode.

3. Define a new subnet. If selected, this option generates the next available subnet, asks for its print name, and enters create mode for the root node of the new subnet.

4. Delete frame. Prompts for a frame number.

5. Move frame. Prompts for source and destination.

6. Utilities. This option goes to a frame which provides a means for reading and writing BH files (the external format for frames).

7. More information. This options enters the rest of the ZED subnet and provides detailed help for all options on the root node and in create and alter modes.

II.2. ZED Create Mode

Create mode provides guidance in building a new frame, while always displaying the current state of the frame being built. It follows the following steps.

1. Abort with error if frame being created already exists.

2. Ask for subnet print name if not already known.

3. Initialize frame by copying default frame for subnet (frame m.0, where m is the subnet number). Will use default frame for whole system (frame 1.0) if subnet default not defined. Set frame identification and store new frame in subnet.

4. Clear screen and display frame being created.

5. Prompt for title, if not already defined (by default). Assumes title position is line 1, column 1. Takes a string terminated by an altmode or carriage-return.

6. Prompt for text, if not already defined (by default). Assumes text position is line 3, column 1. Takes a string terminated by an altmode.

7. Prompt for option, if not already defined (by default). Assumes position with one blank line after last line of text. Prompts with the selector digit (from 1 to 9). Takes a string terminated by an altmode or carriage-return. Prompts for next frame for this option, with default next frame being the next available frame in the subnet. Allows user to user default next frame, no next frame, or specified next frame.

8. Prompt for next option up to option 9. An empty string on an option will terminate the option list.

II.3. ZED Alter Mode

The design for alter mode was derived from extending the one dimensional SOS alter mode (Weiher and Savitzky, 1970) to a two dimensional frame alter mode. The frame is always displayed while it is edited. The user types single character commands (listed below), which are not echoed. The effect of the commands is to alter the frame as displayed. All alterations are made to a copy of the frame, so that the user may abort the edit with no changes made.

When alter mode is entered, a check is made to ensure that the frame is already defined. If it is not, alter mode aborts with an error message. It it is, a copy is made of it, and the copy is displayed. The cursor is set to the first character of the title, which becomes the current item being edited.

Most of the following commands may be preceded by a number (represented by lower case n), which indicates a repetition of the operation. In general, upper case commands are for manipulating items (title, text, options, local pads, global pads), and lower case commands are for manipulating text (within the current item).

General commands:

h or H or ? - Print alter mode help
q or Q - Quit alter mode; no change to frame
e or E - Exit alter mode with altered frame.

Item commands:

n<space> - Skip n characters in current item
n<del> - Back up n characters in current item
l - Back to start of current item
ns<char> - Search for n'th occurrence of <char> in current item

i<char>$ - Insert characters in current item to altmode
nd - Delete n characters in current item
U - Restore current item
nk<char> - Kill (delete) to n'th occurrence of <char> in current item
nm<char><chars>$ - Munch (kill and insert)
t - Transpose next two characters in item
nc<n-chars> - Change next n characters in item

Frame commands:

n<line-feed> - Skip n frame items
n<altmode> - Backup up n frame items
L - Back up to title (first frame item)
S<char> - Search for selection with <char> as selector
I - Insert item; prompts for arguments
D - Delete item

Appendix III. External Frame Format - BH Files

The external frame format was chosen to provide a means of transporting and exporting ZOG frames and nets to other ZOG systems. It was chosen to be compatible with a bibliography maintenance program, called BH (Newcomer, 1976), to allow use of the sorting features of BH if desired. A BH file contains a series of entries with each entry containing a number of elements. The first element must be a +A+ element. The other elements are all labelled with +<char>+. The following BH compatible format is used for ZOG frames. See Figure 18 for an example of its use.

+A+ m.n v - Entry header. Define frame n in subnet m with version number v. If v is less than or equal to an existing version number, then bypass this old version. If v is zero, then augment the existing frame rather than replacing it.

+A+ -n - Alternate entry header. Define global pad n.

+A+ 0 - Alternate entry header. No frame or pad defined. This entry is used to specify information global to a series of entries, like comments or subnet descriptors.

+G+ n - Global pad index. This specifies that the frame being defined should include the specified global pad. BH allows the use of & to repeat the last +<char>+ element, so +G+ n1 & n2 & n3 & ... & nm can be used to specify a series of global pads for a frame.

+N+ m "name" - Subnet descriptor. This specifies that subnet m can be referenced by the specified print name. Subnet descriptors should appear in a +A+ 0 entry before the first frame definition for that subnet. This is necessary because the frame identification, which is constructed when the frame is defined, is built from the frame number and the subnet print name.

+C+ "comment" - Comment. If a frame is being defined, the comment is included as part of the frame, although it is not normally displayed. Otherwise, the comment is ignored.

+F+ <desc> m.n - Next frame number. This specifies the next frame for either an option or a pad. <desc> is described below.

+P+ <desc> v h - Position. This specifies the cursor starting position (vertical and horizontal) for an option or pad.

+T+ <desc> "text" - Text. This specifies the text which is displayed as part of an option or pad. It should include the selector character and some indication of whether making the selection would result in any action or next frame (i.e., if not, the selector character should be preceded by a "-").

+X+ <desc> "action" - Action. This specifies the character string which will be used as action input when the option or pad is selected.

+F+, +P+, +T+, and +X+ are used to characterize a selection (option or pad). Any of them may be omitted. <desc> specifies which option or pad is being characterized. <desc> = T - frame title (for +P+ and +T+ only) F - frame text (for +P+ or +T+) - frame action (for +X+) - illegal for +F+ O<char> - option with selector char. L<char> - local pad with selector char. G<char> - global pad (only if defining a global pad)

Text appearing between double quotes may be multiple lines long. "" will force a single " to be included in the text. Because of BH limitations, + and & must be entered as ++ and &&.

Appendix IV. Terminology

The following is an alphabetized list of terms used throughout the paper with a brief definition for each.

accessor - If frame A has a selection whose next frame is frame B, then frame A is said to be an accessor to frame B. Each frame has a list of all its accessors for maintenance purposes.

action - An action is a text string which may appear in a selection or a frame. The interpretation of an action is to send the characters to the set of logical output devices linked to the "action" logical input device. Escape characters present in the text will invoke the communications language.

BH - A bibliography system (Newcomer, 1976).

BH format - A BH compatible format which we use for an external representation of frames.

communications language - A language which can be invoked through escape characters ↑A, ↑B, or ↑D from keyboard, input file, subjob, or actions. It supports ZOGNET positioning, communications control, and ZOGNET modification.

communications routing - Each logical input device is routed to a set of logical output devices. The communications language can control this routing information.

context display - A window near the upper right corner of the display area which is used to show the frame identification of the last marked frame. This is the frame to which a return would return.

C.mmp - A multi-mini-processor being developed at Carnegie-Mellon University. The rapid-response version of ZOG will reside on this computer.

escape character - A distinguished character which forces a special interpretation on the next character. In ZOG, the characters ↑A, ↑B, and ↑D are escape characters to invoke communications language commands.

frame - The basic display unit. A frame contains some text and a menu of selections, and may contain an action to be interpreted on entry.

frame action - The action to be interpreted on entry to a frame.

frame comment - Text which may be included in a frame, but which is not normally displayed.

frame display - The display area used for showing the frame.

frame identification - The print name for a frame. It is normally the subnet name followed by the relative frame number (e.g., ZOG36). It is always displayed in the upper right corner of the display.

frame library - The total set of frames which exist. The total collection of existing BH files.

frame number - The subnet index and relative frame number. The frame number is normally in the form "m.n", meaning frame "n" in subnet "m".

frame text - A character string included in a frame and displayed when the frame is displayed.

frame title - A character string included in a frame and displayed when the frame is displayed. By convention, the title is one line long and is displayed on the top line of the display.

global pad - A pad which is common to a large part of the ZOGNET. Global pads are stored globally and are accessed through global pad indices.

global pad index - The number which is used to access a global pad.

Graphics Display Processor - A vector drawing graphics system developed at Carnegie-Mellon University. This system is one of the primary components in a ZOG terminal.

IMS disk - A secondary storage device used on C.mmp for paging. This device has page transfer feature which allows zero latency access to pages. It is one of the primary components in a ZOG terminal.

large-network - A large set of frames is necessary in a rapid-response system. The PROMIS system has approximately 40,000 frames.

local pad - A pad which is local to a frame. These normally appear in the right column of the display and are selected with upper case alphabetics. They are distinguished from options only by the next ZOGNET positioning command.

logical input device - A source of input for ZOG. Currently, one of: input file, keyboard, touch screen, actions, ZOGNET printing, subjob, or echo from subjob.

logical output device - A destination for characters being processed by ZOG. Currently, one of: null, selection processor, ZOGNET builder, subjob, output file, user display, frame display, or context display.

L* - An interactive system-building system developed at Carnegie-Mellon University (Newell, Freeman, McCracken, and Robertson, 1970). The language used to implement ZOG.

menu selection - The technique of displaying a set of alternatives and allowing the

user to select from that set. This technique has a long history of use for man-machine communication.

network - A collection of frames.

option - A selection which normally conveys information and, if selected, moves the user to a new frame. Options are displayed in the middle of the display area and are selected with digits.

pad - A selection which normally performs some action, and may or may not move the user to a new frame. There are two kinds of pads, local and global.

PDP10 - A 36-bit word computer. The current implementation of ZOG is on this machine.

PROMIS - A medical information system being developed at the University of Vermont (Hurst and Walker, 1972). The PROMIS system is the first system to use a rapid response, large network, menu selection communication philosophy.

rapid-response - When a selection is made, the next frame should be displayed rapidly enough that it appears instantaneous to the user. In practice, this means less that 1/4 second 70% of the time.

relative frame number - Each frame belongs to a subnet and is numbered relative to that subnet. For example, ZOG36 is the 36'th frame in the ZOG subnet, and its relative frame number is 36.

selection - A part of a frame which is displayed as part of a menu and may be selected by the user. A selection is either an option, a local pad, or a global pad.

selection action - An action which associated with a selection. It is interpreted if the selection is selected.

selection processing - The part of ZOG which takes a character and selects an option, local pad, or global which is then evaluated. To evaluate a selection, the selection processor interprets the selection action, if any, then gets the next frame, if any, then displays the frame, if new, then interprets the frame action, if new.

selector character - A character stored as part of a selection and used by the selection processor.

SOS - A line oriented text editor developed at Stanford (Weiher and Savitzky, 1970).

subjob - A second job under the control of the ZOG job. It may be used to run an arbitrary program. Communication to it is through the subjob logical input and logical output devices.

subnet - A collection of frames with a common purpose. Frames within subnet are

accessed relative to the subnet. A subnet is referenced by an index and may have a print name associated with it.

subnet index - The number used to refer to a subnet.

subnet name - The print name associated with a subnet.

subnet root node - Relative frame one in a subnet. Entry to and subnet is normally through its root node.

touch screen - A clear glass screen placed over a display terminal, which responds to the touch of a human finger with a pair of coordinates.

user display - The display area reserved for arbitrary user use. On terminals which have small screens, this can be anywhere on the display, determined by the location of the cursor. On large screen graphics, this is a separate window.

user profile - A user defined set of modifications to the system which tailors the system to the users needs. In ZOG, this is accomplished by allowing users to edit frames and maintain their own copies.

version number - Each frame has a version number stored with it to aid in maintenance.

ZOG - A man-machine interface. The original ZOG was developed at Carnegie-Mellon University in 1973 (Newell, Simon, Hayes, and Gregg, 1972), and stressed the guidance aspects of the interface. The latest ZOG, ZOG2, is the topic of this paper. It stresses rapid-response and large-network for guidance and other applications.

ZOG-like system - A system which utilizes rapid-response, large-network, menu selection for man-machine communication.

ZOGNET - The set of frames which the user can access through ZOG.

ZOG terminal - A specialized terminal designed for rapid response. This includes a Graphics Display Processor, an IMS disk, and a touch screen.

ZED - A frame editor. This system includes three parts: a create mode, an alter mode, and a ZED subnet for deleting and moving frames.

References

Chomsky, N., Syntactic Structures, The Hague: Mouton, 1957.

Heart, F., R. Kahn, S. Ornstein, W. Crowther, and D. Walden, The Interface Message Processor for the ARPA Computer Networks, Proc. AFIPS 1970 SJCC, Vol. 36, AFIPS Press, Montvale, N.J., pp. 551-567.

Hurst, J. and K. Walker (eds.), The Problem-Oriented System, MEDCOM Press, New York, 1972.

Martin, J., Design of Man-Computer Dialogues, Prentice-Hall, 1973.

Newcomer, J., BH - A General Information Organization Program, Carnegie-Mellon University Technical Report, May 1976.

Newell, A., Notes for a Model of Human Performance in ZOG, Carnegie-Mellon University Technical Report, August 1977.

Newell, A., H. Simon, R. Hayes, and L. Gregg, Report on a Workshop in New Techniques in Cognitive Research", Carnegie-Mellon University Technical Report, June 1972.

Newell, A., P. Freeman, D. McCracken, and G. Robertson, The Kernel Approach to Building Software Systems, 1970-71 Computer Science Research Review, Carnegie-Mellon University.

Rosen, B., Graphics Display Processor Programmer Guide, Carnegie-Mellon University Technical Report, January 1974.

Sutherland, I., SKETCHPAD: A man-machine graphical communication system, Proc. AFIPS 1963 SJCC, Spartan.

Weiher, W., and S. Savitzky, Son of Stopgap (SOS), Stanford Artificial Intelligence Laboratory Operating Note, 1970.

Wulf, W., and C. Bell, C.mmp -- A Multi-Mini-Processor, Proc. AFIPS 1972 FJCC, Vol. 41, AFIPS Press, Montvale, N.J., pp. 765-777.

Wulf, W., C. Pierson, F. Pollack, R. Levin, W. Corwin, A. Jones, and E. Cohen, Hydra: The Kernel of a Multiprocessor Operating System, CACM, Vol. 17, no. 6, June 1974, pp. 337-345.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: ZOG: A MAN-MACHINE COMMUNICATION PHILOSOPHY	5. TYPE OF REPORT & PERIOD COVERED: Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): G. Robertson, A. Newell, and K. Ramakrishna	8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0874 F44620-73-C-0074
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University Computer Science Dept. Pittsburgh, PA 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research Arlington, VA 22217	12. REPORT DATE: August 4, 1977 13. NUMBER OF PAGES: 57
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

None

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*