COMPUTER SCIENCE & TECHNOLOGY:

A TESTBED FOR PROVIDING UNIFORMITY TO USER-COMPUTER INTERACTION LANGUAGES

NBS Special Publication 500-63

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

COMPUTER SCIENCE & TECHNOLOGY:

A TESTBED FOR PROVIDING UNIFORMITY TO USER-COMPUTER INTERACTION LANGUAGES

Siegfried Treu

Center for Computer Systems Engineering
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick Secretary

Luther H. Hodges, Jr., Deputy Secretary

Jordan J. Baruch, Assistant Secretary for Productivity, Technology and Innovation

U.S. NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued August 1980

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested pecialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publicaton.

National Bureau of Standards Special Publication 500-63
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-63, 74 pages (Aug. 1980)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 80-600107

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1980

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Price $4.00
(Add 25 percent for other than U.S. mailing).

A Testbed for Providing Uniformity to User-Computer Interaction Languages

Siegfried Treu
Institute for Computer Sciences and Technology
National Bureau of Standards
and
Department of Computer Science
University of Pittsburgh

The differing user-computer interaction languages, implemented for conducting the same applications-specific functions on different systems, represent significant stumbling blocks to users. Toward alleviating this problem area, the use of an intermediary processor to "uniformize" interaction languages is presented. A framework for such processing is characterized in terms of the required intermediary actions and the logical capabilities needed to perform those actions. The testbed software facilities, centered on the NBS Network Access Machine, are then portrayed. Throughout, an example application, using a common command language subset to access five bibliographic retrieval systems, is described.

Key words: Bibliographic retrieval systems, command language, interaction language, language transformation, language uniformity, user-computer interaction, user-oriented system design.

A TESTBED FOR PROVIDING UNIFORMITY TO
USER-COMPUTER INTERACTION LANGUAGES

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
1.1	Purpose and Overview	1
1.2	Interfacing Users and Computers	2
1.3	Interaction Language	3
1.3.1	Motivating Example	3
1.3.2	Accommodating Users	4
1.3.3	Literature Review	5
2.	STUDY APPROACH	6
2.1	Multi-System Orientation	6
2.2	Intermediary Uniformizer	7
2.3	Selected Application	8
2.3.1	Bibliographic Retrieval	8
2.3.2	Operational Systems	9
2.3.3	Retrieval Functions	9
3.	TRANSACTION PROCESSING	10
3.1	Design Framework	12
3.2	Intermediary Actions	12
3.3	Required Capabilities	15
3.3.1	Matching of Character Strings	17
3.3.2	Selection of Components	18
3.3.3	(Re) construction of Messages	19
3.3.4	Mapping of Message Sequences	20
3.3.5	Handling of Contingencies	21
4.	TESTBED FACILITIES	21
4.1	Network Access Machine	21
4.2	Communication Modes	22
4.3	NAM Command Repertoire	23
4.3.1	UNIX Commands	23
4.3.2	Macro Directives	23
4.3.3	Conditional Constructs	25
4.3.4	String Manipulation	27
4.3.5	Macros	29
4.4	Other Features	30

Page

5. LANGUAGE DESIGN 30

5.1 Criteria for Command Language Design 30
5.2 Command Language Subset 32
5.3 Implemented Macros 35
5.4 Demonstration and Testing 35

6. CONCLUSIONS 35

REFERENCES 37
APPENDIX A Command Macros A1
APPENDIX B Example Session Transcripts B1

1. INTRODUCTION

A considerable diversity of hardware and software has been created by manufacturers of computer equipment and by those who, in utilizing that equipment, provide various kinds of applications-oriented services. That diversity is becoming increasingly troublesome to the computer users or service consumers who are forced to cope with the resulting system inconsistencies and incongruities.

1.1 Purpose and Overview

This paper is to present a framework for performing studies of user-computer interaction languages. It is based on a testbed facility developed and available at the National Bureau of Standards. More specifically, the framework encompasses the ability to focus on a set of systems (called object systems) designed to carry out essentially identical applications- specific functions. Thus, for example, we can consider "text editing" language design with regard to a set of different text editors available; or we can study the language for accomplishing FORTRAN program compilation by accessing a set of different FORTRAN compilers available. Each such set exhibits similarities in function but differences in associated user-computer interaction language.

For the particular demonstration of our testbed facilities, a set of five on-line bibliographic retrieval systems was employed. Utilizing the NBS Network Access Machine as an intermediary processor or translator of the interaction languages involved, the differing implementations of command language were "uniformized" into a common language. This approach and the actions and capabilities required to carry it out are described in Sections 2 and 3 respectively. Section 4 then presents an overview of NBS testbed facilities which are able to meet those requirements. Finally, the considerations in designing our example common command language (for bibliographic retrieval), to be employed on the user side of the uniformizing intermediary processor, are described in Section 5 with references to detailed examples in the appendixes. Several conclusions from this work are outlined in Section 6.

It should be pointed out that the testbed facilities are as yet limited to application to the kind of well-defined, homogeneous set of command languages mentioned above. However, they have the potential for considerable expansion and improvement. Furthermore, the particular command language subset suggested and used as a command language for bibliographic retrieval (in Section 5) is not being proposed as a "standard" language. It is instead to be viewed as an example version which could be tested, according to various criteria, against other proposed

languages. As a result of further research and testing, hopefully producing meaningful measures and criteria for evaluating interaction languages, it may then become possible to develop user-oriented standards.

1.2 Interfacing Users and Computers

The design of the interface between users, or human information processors, and their artificial counterparts, namely computers, is a very difficult task. That difficulty was undoubtedly a major factor in the relative neglect of user needs until fairly recently. However, professionals in computer science and related disciplines are now becoming increasingly sensitive to the importance of user-oriented systems design.

Evidence to corroborate an increase in user orientation includes the enlarging body of pertinent literature (e.g., Bennett [1972], Foley and Wallace [1974], and Rouse [1975]); the activities of certain major special interest groups, e.g., the ACM/SIGGRAPH [Treu, 1977a]; and sponsorship of international forums such as the NATO Advanced Studies Institute on Man-Computer Interaction, conducted in Greece in September 1976. It is also evident that the Federal Government, through agencies such as the National Bureau of Standards, is actively fostering and pursuing user-oriented approaches to computer systems design, service, and evaluation. That is born out by this as well as a considerable series of previous reports and papers (e.g., Pyke [1973], Blanc [1974], Abrams and Cotton [1975], Rosenthal [1976a], Abrams and Treu [1977], Mamrak and Amer [1979]).

In order to achieve a user-oriented computer interface, numerous considerations must in general be taken into account (e.g., Treu [1977b]). They range from psychological/physiological questions about the user to software/hardware concerns about the computer. In this report, the emphasis is only on one constituent of the interface: the user-computer interaction language encompassing both the user commands and the system responses. However, that is probably the most significant constituent. The objective in this project is to overcome some of the previously mentioned diversity in computer service implementations by providing the user with a special type of "uniformity" in interaction language.

1.3 Interaction Language

The average American traveller appreciates the fact that he or she can speak English in many foreign countries, instead of having to learn French, German, Japanese, etc. Likewise, the average computer user surely would prefer to speak one interaction language to accomplish essentially the same tasks in each of a variety of different computer systems.

1.3.1 Motivating Example

Suppose that a user has become familiar with a particular interactive bibliographic retrieval system from which the following command,

FIND Subject Term A AND Subject Term B [CR]

will elicit a response giving the number of bibliographic citations in the system's database which are concerned with the logical conjunction of subject terms A and B. In responding to the above search command, the system also assigns a sequential search statement number to it. The user can thereby make later reference to that statement within the context of a subsequently entered search command. For example, if the assigned statement number is N, the user could use it in a compounded search request such as:

FIND N AND Subject Term C [CR]

If that same user now accesses a certain other interactive bibliographic retrieval system, the very same search objective involving the same kind of database has to be accomplished as follows:

SELECT Subject Term A [CR]

succeeded by a system response, giving the number of relevant citations and assigning a sequential reference number, say m. The user must then proceed with:

SELECT Subject Term B [CR]

and expect another system response, including reference number n. Finally, the user must issue the command:

COMBINE m AND n [CR]

to arrive at the same kind of system response which was produced with the first system after only entering the FIND command.

This example is only one illustration of the inconsistencies in language that have resulted from differing implementations even when identical applications (e.g. bibliographic retrieval) are involved. Such inconsistencies become even more cumbersome and confusing to users when more than two applications- specific systems are considered. They are confounded still further when situated in more general, multi-purpose systems.

1.3.2 Accommodating Users

We are led to ask some pertinent questions about interaction language design: Whose needs should be accommodated, the computer's or the user's? Should we attempt to standardize interaction language or is some kind of compromise solution available, providing at least partial relief for users without imposing language standards on anyone?

While sociologists, psychologists, linguists, and educators may study and explore ways of improving communication among people, including various modes of interpersonal collaboration [Chapanis, 1975], any attempts to influence communication means and methods are distinctly constrained by the fundamental fact that human information processors are so individally different and relatively uncontrollable. However, the latter is not true of computers. Computer systems can be controlled and modified in their behaviors even though, like users, they may be individually different as a result of differing implementations.

In view of the above, it seems reasonable that, if given the choice between already experienced (or conditioned) users and existing (or operational) computers, we should modify the computers to be suitable for communication by users rather than vice versa. Few users who have acquired a distinct dialect for talking with a certain computer are likely to want to learn a markedly different dialect in trying to access other systems serving similar or identical functions.

On the other hand, while the computers may be willing to accommodate different user dialects, their managers may not be. For a variety of reasons, perhaps based on their views or analyses of the competitive marketplace, they may resist changing their system, e.g., in its command language. As a consequence, we continue to have new users learn and become conditioned to the special interaction language dialect pertaining to one system. Then, when motivated to access a similar resource on a different system, they find they can't speak its language.

The project reported here represents an attempt to enable language uniformity, at least for categories of applicatons-specific systems. Before describing our particular approach, a brief look at the literature concerned with interaction language is presented.

1.3.3 Literature Review

The literature exhibits a few concerted attempts to address what are termed "command languages." One of these involved a small, international conference which resulted in the publication of a set of papers, including a review of the background of command languages [Enslow, 1975]. Another effort was due to the Special Interest Group for Operating Systems (SIGOPS), which conducted a session entitled "Operating System Command Languages" at the 1976 Annual Conference of the ACM. The four resulting papers, published in the proceedings, ranged from a rather conceptual view of the user- computer interface [Cheriton, 1976] to a specific implementation of a particular command language [Muchnick, 1976].

As demonstrated by the above-referenced papers, concern about command languages has generally been synonymous with asking about the types of facilities and features available to a user in "commanding" the supervisory or operating system to carry out various functions or to get at available compilers, editors, and other software and data processors. This mostly one-sided, monitor-level view, motivated by the necessity of having to use certain commands to access system resources, without really attempting to perform a comprehensive study of the interaction language (including system responses to user commands), is indicative of the literature background. There have, of course, been suggestions on how command languages might be improved, and dismay has been expressed about the numerous inconsistencies and incongruities, but the study of user-computer interaction language theory and design, has been lacking.

Besides the above, various special-purpose command languages for utilizing particular kinds of application packages or software processors, have received some attention. Examples are found in selected papers in the proceedings of the 1976 ACM Symposium on Graphic Languages [Berk, 1976] and a report on utilizing a common command language for file manipulation [Fitzgerald, 1978]. However, unlike programming languages which have been studied to considerable extents, and for which various language constructs and standards have been developed over the years, command languages have, in general, been neglected.

2. STUDY APPROACH

In the interest of accommodating user needs, without imposing the requirement of interaction language standardization on the computer system designers and service providers, it is possible to achieve a reasonable compromise: Enable the multi-system user to employ a single interaction language by means of an intermediary "uniformizing" processor, at least for a particular category of applications-specific systems.

2.1 Multi-System Orientation

As already implied by the above, the focus of the work described here is not on modification of any one system's interaction language for the sake of its own clientele. The latter may be quite satisfied with what they have, especially if they have not tried other, perhaps better versions. Concern is instead for those experienced users who do want to access a variety of systems presenting similar (as well as different) resources. Concern is also for the increasingly sophisticated new users who, unlike their predecessor generation of users, are quite aware of the multiplicity of systems available and are interested in shopping around and "playing the field." They are no longer satisfied with having only one system to work with.

This multi-system orientation suggests several possible kinds of solution. The following three must be rejected as undesirable or currently infeasible:

1. The users should be required to learn the different interaction language dialects. This would be contrary to the theme of this study which is user-oriented design of interaction language. A single language usable on a set of different systems (with similar functions) is certainly easier on a user than a variety of inconsistent languages required for the same set of systems.

2. The object systems should be required to adopt and implement some standard interaction language, designed for a particular type of application or system category. This choice must be rejected because we do not have appropriate standards developed as yet.

3. The object systems should be required to understand (i.e. translate) all the different language dialects of different groups (and possibly even individual) users. This option is not feasible because of the tremendous overhead processing that would be incumbent on the object systems and, furthermore, considerably more research into

language-based adaptation (by the system to the user) is prerequisite to effective implementation of this capability.

Thus, we are left with the following possible solutions:

4. An intermediary system should be used to make it appear (to the users) that solution (2) has been applied, without really imposing on the object systems (or their management).

5. The intermediary system should be used to make it appear (to the users) that solution (3) has been applied, again without really imposing on the object systems.

Solution (4) seems to be a viable alternative which, if properly carried out, could please both the users and the systems' managers. Furthermore, it would have the special by-product of enabling the testing of proposed standard languages and consequently of perhaps convincing management of the desirability of language standardization and/or adaptability.

The fifth solution is in a sense subsumed by (4), but only in part. Depending on the power and versatility of the intermediary processor, it could at least provide the testbed for a variety of interaction languages, thereby adapting to respective preferences of different user groups. Although some related work with "intelligent terminals" providing for limited "learning" capabilities has been carried out [Anderson and Gillogly, 1976], it remains premature and, given the state-of-the-art of "adaptive" systems, unrealistic to suggest that such an intermediary processor can be designed in the near future to be truly adaptive to individual user language preferences. However, some constructive steps in that direction are being planned at NBS.

2.2 Intermediary Uniformizer

In accordance with the above-selected alternative (solution (4)), the user's multi-system interaction is to be facilitated by means of an intermediary system which we shall call a "uniformizer." That is to say, the intermediary system gives the appearance to the user that a particular interaction language, applicable to a particular level of or application in the object system's resources repertoire, pertains uniformly to each of a number of different object systems.

The intermediary uniformizer should ideally be totally transparent to the user. However, that transparency is currently not attainable in view of some system discrepancies or inconsistencies which are difficult if not impossible to reconcile [Markus and Reintjes, 1976] and "uniformize." Hence the intermediary system may at times have to take exceptional or default actions which will inevitably reveal its existence to the user. Such problems are not to deter the objective of providing for as much uniformity as is reasonably possible.

2.3 Selected Application

Our intention is to present a general framework for design and testing of user-oriented interaction language which may be applicable to a number of different types of interactive system resources. To demonstrate the requirements and feasibility of utilizing our testbed facilities (to be described in the next sections) for purposes of "uniformizing" language, a particular kind of application was selected.

2.3.1 Bibliographic Retrieval

It was decided that the area of on-line information retrieval, i.e. retrieval of bibliographic citations, would be a good candidate. Reasons included the fact that features of the most prominent of such systems have been well described [Martin, 1974] both in terms of commonalities and differences. Systems of that type are also reasonably self-contained and well-defined in terms of applicable retrieval or retrieval-related functions and hence in required user-issued commands.

Nevertheless, in spite of the functional similarities, there are enough differences among bibliographic information retrieval languages and their implementations to warrant the facilitation of user access through a common command language. This fact is confirmed both by Martin [1974], who suggested that investigation of uniformity in commands be carried out, and also by the NSF-sponsored study by Marcus and Reintjes [1976] at M.I.T., involving "coupling" of retrieval systems. The differences among the user-issued, retrieval-related commands generally fall into three categories (and combinations thereof):

1. Semantic: Two different command verbs used to achieve essentially the same results in two different systems (e.g. **NEIGHBOR** and **EXPAND** to output related vocabulary terms).

2. Syntactic: Different sequencing or delimitting of command components within similar commands used on different systems.

3. Scenario: Beyond one-to-one distinctions between similar commands, the differences between scenarios of commands used on different systems to accomplish the same functions. An example was given in Section 1.2.1.

A similar characterization can also be made of the differences in the responses from different systems. Our primary attention in this study was focussed on overcoming the discrepancies in user-input commands.

2.3.2 Operational Systems

Fortunately a number of operational systems providing useful bibliographic retrieval services were made available. Five systems were selected. In alphabetical order, they are:

1. BASIS
   -Battelle Memorial Institute
2. DIALOG
   -Lockheed
3. MEDLINE
   -National Library of Medicine
4. ORBIT
   -Systems Development Corporation
   -Close relative of MEDLINE
5. RECON
   -Energy Research and Development Administration
   -Relative of DIALOG

All of these systems were accessible via the Tymshare Network in case the same communications medium was to be employed. However, the testbed facilities are not constrained to any particular communication setup or facility. Two of the above systems, ORBIT and DIALOG, will be referenced almost exclusively in this report for purposes of illustrative examples.

2.3.3 Retrieval Functions

Each of the sample set of five systems serves the need of storing and then enabling users to retrieve bibliographic citations in various specialty areas and in response to user-specified search logic. Although their files may be different, the substantive functions to be carried out are conceptually quite similar.

The major substantive functions are listed in the left column of Figure 1: The user can

1. Request identification of all the files or databases that are searchable in a given system;

2. Select the particular file of interest;

3. Ask for a display of the search term vocabulary, which may be structured alphabetically and/or hierarchically in different systems;

4. Specify and initiate a search of the selected file, based on a logical (Boolean) combination of search terms (from the vocabulary) and pertinent qualifiers;

5. Based on a search (4 above), request the output of a specified number of responding citations, in accordance with some output format.

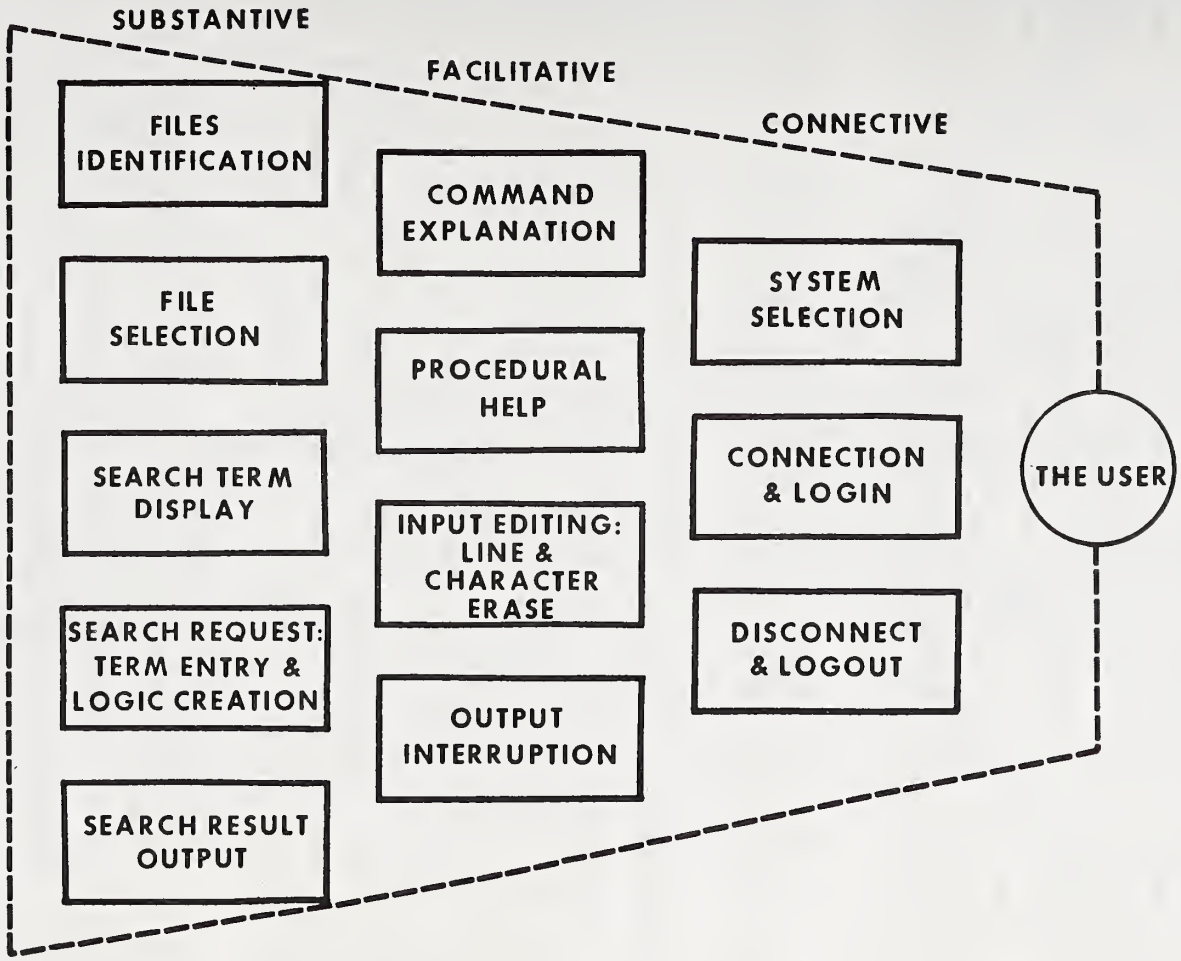
Besides the substantive functions, similarity also exists in the facilitative and connective types of functions, which are far less applications-dependent. As shown in Figure 1, a user must be able to identify the desired retrieval system, connect and login and later, after completing an on-line session, logout again. In the meantime, various facilitative capabilities, involving explanatory and procedural help as well as administrative assistance (input editing and output interruption) should be provided.

Figure 1 thus presents a capsule view of the subset of retrieval-related functions selected for this study. Ideally, in an application of this kind in which user-perceived uniformity is to be provided across a number of systems, one would like to include all possible functions and design a correspondingly all-inclusive command language. However, due to the well-established fact that different system implementations can render certain function-specific commands irreconcilable, only a major subset of functions was addressed. It was deemed to be realistic and powerful enough to make the experimental application productive within a resonably short time period.

3. TRANSACTION PROCESSING

Because software and hardware implementations tend to be constrained by particular language characteristics and hardware features, the intermediary processor which plays our language uniformizer role should first be defined in terms of functional actions and capabilities. The more

FIGURE 1. SUBSET OF RETRIEVAL-RELATED FUNCTIONS



general framework for intermediary processing of the user-system transactions can thereby be portrayed. The practical instantiation of that framework using our testbed facilities can subsequently be described more meaningfully (in Section 4).

3.1 Design Framework

Up to this point, we have advocated user-oriented design of the language interface by means of employing an intermediary processor to facilitate the interaction between the user and any one of a set of accessible systems. This transaction model is depicted in Figure 2. The intermediary processor is to uniformize the different interaction languages, implemented for the object systems respectively, by enabling the user to utilize one common language for all of them. Since the application for which this intermediary processing approach was to be demonstrated is bibliographic information retrieval, the design of a suitable common retrieval language was part of our effort. The criteria for doing so are discussed in Section 5.

In this Section, the design of the intermediary system, i.e. the uniformizer, is considered in terms of the required functional actions and capabilities.

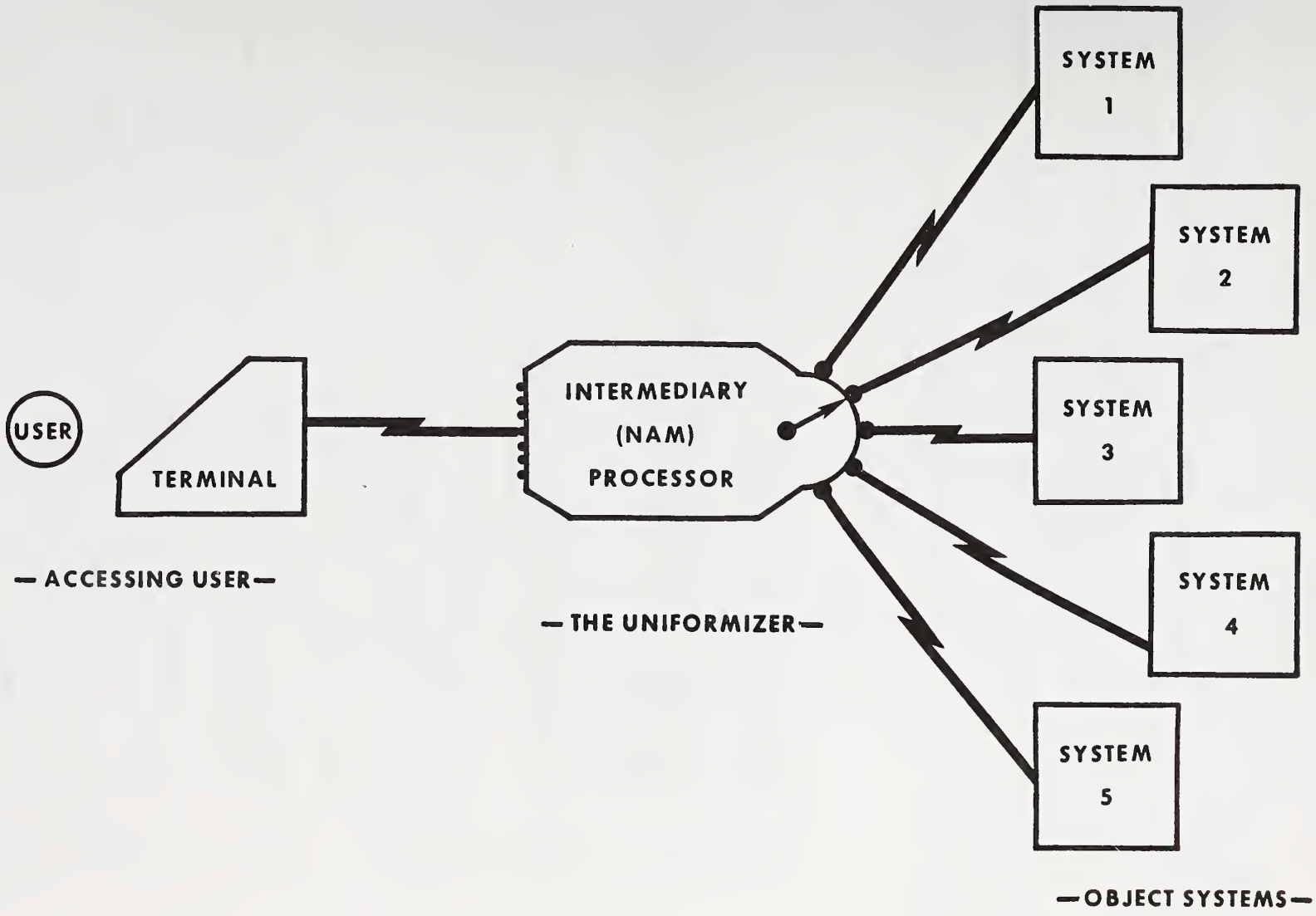
3.2 Intermediary Actions

Assuming that the user has already gained access to the selected object system, via a brief interchange with the uniformizer (see Figure 2), and is now working at the regular user-system transaction level involving some monitor-level or applications-specific software processor, the user imputs and system outputs can be distinguished as follows: the user either commands the system to do something, explicitly or implicitly (by asking a question) or the user responds to the system by answering specific questions (e.g. input of parameter value or choice of available options).

The "system," on the other hand, which is presently used with reference to either the intermediary or the object system, reacts to inputs (consisting of above- indicated commands or answers) by either accomplishing the required processing (if possible), sending appropriate messages and/or requesting additional input.

Given the general possibilities and distinctions suggested above, we can now characterize the actions required of the uniformizer in carrying out its intermediary role between the user and the remote, object system:

FIGURE 2. UNIFORMIZED USER-SYSTEMS INTERACTION



1. The user issues a command or answer intended for the object (sub)system (but possibly needed by the uniformizer); Go to Step 2.

2. The uniformizer intercepts the command or the answer, parses it, and takes one of the following actions:

   a. Forwards it unchanged to the object system; Go to step 3.

   b. Constructs revised command and transmits it to the object system; Go to Step 3.

   c. Reacts to the user by transmitting a message which either

      (1) Requires additional or alternate input; Go to Step 1: or

      (2) Is a terminal/default message; Go to Step 1.

3. The object system receives the command or the answer and carries out one of the following:

   a. Performs the intended function and transmits result or acknowledgement to the user or possibly only the uniformizer which may need it; Go to Step 4.

   b. Declines or fails to perform the intended function due to one or more of
      (1) Lack of capability

      (2) Inaccurate input parameter

      (3) Error condition

   and transmits an identifiable (by the uniformizer) message back to the user; Go to Step 4.

4. The uniformizer intercepts the system response or message, parses it, and takes one of the following actions:

   a. Forwards it unchanged to the user; Go to Step 1.

   b. Constructs revised system response and transmits it to the user; Go to Step 1.

   c. Reacts to the system by transmitting a command which either

(1) Requires additional result; Go to Step 3; or

(2) Requests clarification of problem or error condition; Go to Step 3.

These functional steps can be portrayed graphically by means of the flowchart of Figure 3. Two particular characteristics become apparent: first, the inherent symmetry in the treatments by the uniformizer of the user input and system output respectively; secondly, within that symmetry, the interactive nature of user-uniformizer communication until a command is ready for transmission to the system, and likewise of system-uniformizer communication until a response is ready for transmisson to the user. Notice that, ideally, the uniformizer should also be capable of deciding whether a user input is legal or safe for (understandable reaction by) the system and whether, on the other hand, a system response (e.g., error message) is adequately clear for the user.

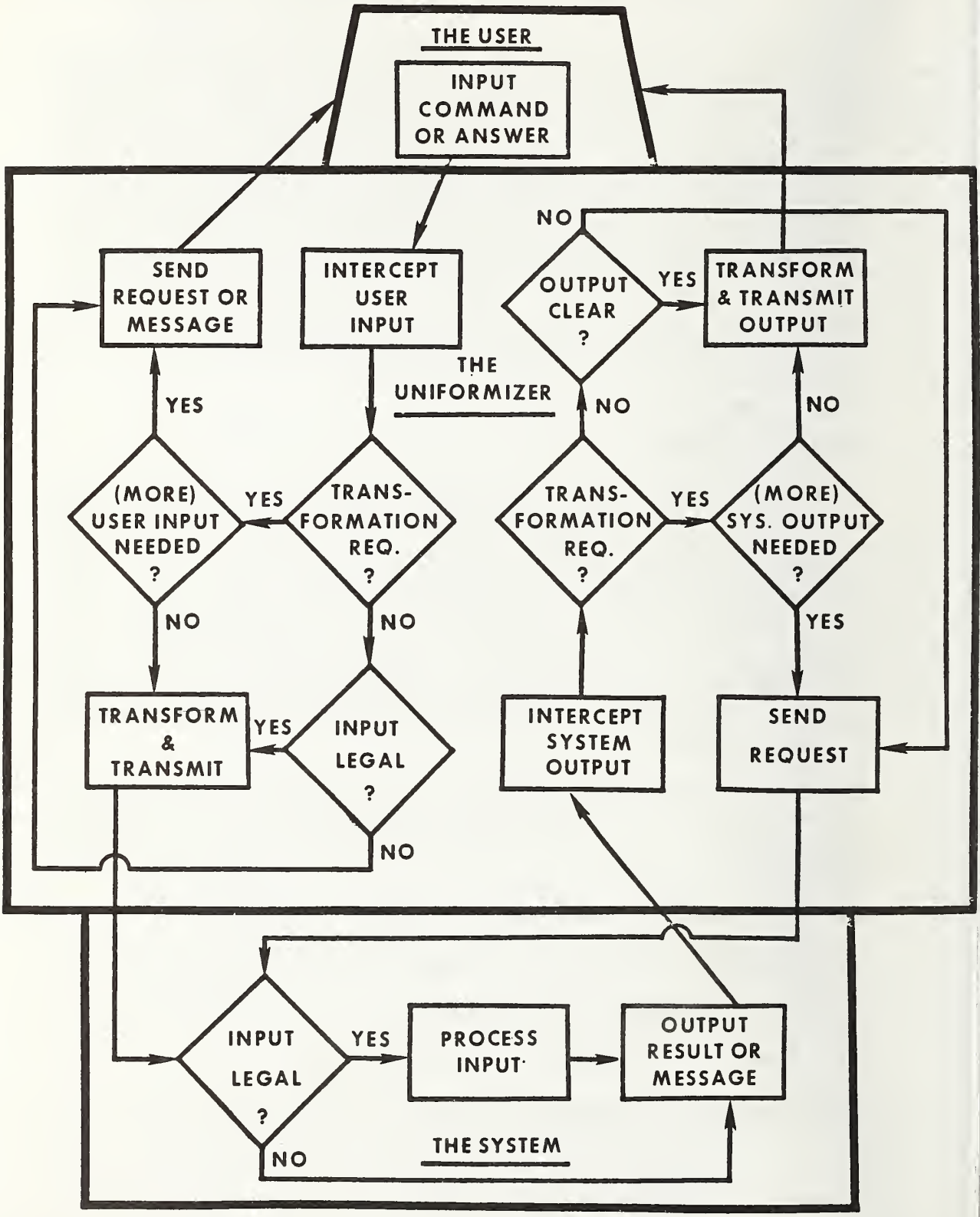
3.3 Required Capabilities

In order to enable the above-indicated intermediary actions, the uniformizer must not only have the communications-related capabilities of receiving messages from and transmitting messages to each of the other processors but it must also be prepared to carry out any required processing of messages in its possession at any given time. The latter necessitates the development and implementation of appropriate algorithms so that the "parsing" of messages or the "construction" of revised commands or the other actions suggested in Section 3.2 can in fact take place.

The following five types of fundamental capabilities seem to be required of the uniformizer to enable it to carry out the essential transaction processing and message transformation:

1. Matching of character strings,
2. Selection of message components,
3. (Re)construction of messages,
4. Mapping of message sequences, and
5. Handling of contingencies.

FIGURE 3. UNIFORMIZER PROCESSING OF INPUT/OUTPUT



The above are interdependent or mutually supportive in that they (or a subset of them) may have to be employed in combination or in immediate succession. As can be observed from their respective descriptions in the following subsections, these capabilities pertain specifically to message processing under the assumptions that

1. The messages are constituents of one or more clearly defined artificial interaction language(s) thereby avoiding most of the problems of dealing with natural language parsing and grammatical manipulation, and

2. An adequate body of knowledge, or enough "intelligence," resides in the intermediary system to support the required processing of such languages.

In each of the descriptions that follow, an illustrative example is given based on transactions or messages involved in our selected bibliographic retrieval application.

3.3.1 Matching of Character Strings

Given the following conditions:

1. The uniformizer has received any message, i.e., character string, consisting of n alphanumeric characters, for $n > 0$, and

2. A reference point i has been established, either explicitly or implicitly during interprocessor interaction, to focus on the substring starting with the ith character in the message, for $1 \leq i \leq n$,

the system must be able to effectively determine matches or mismatches between that message substring and another character string which is either already stored in its resident knowledge base or explicitly supplied by one of the two communicating processors.

Subsequent uniformizer actions are then contingent on the outcome of the specific character string matching. This fundamental capability clearly suggests that the intermediary system employed for this role should be equipped with software which facilitates efficient character string storage and manipulation.

Example: When the user wishes to command the

system to display one of several different types of available information, e.g. a list of names of searchable files or a list of search terms related to some specified term or a specified number of responsive citations, then the following "common" command format is suitable in all cases.

DISPLAY <Argument(s)> [CR]

Upon interception of this message, the first several characters (starting at i=1) must be matched against known types of commands to identify the command category involved. Then, at the hierarchically next level, the argument character string (starting at i=9 in above command) must be matched against the known options that apply to the DISPLAY command. The uniformizer can then take the action(s) required by the user-specified option in order to transform the message appropriately.

3.3.2 Selection of Components

Given a more macro view of a communicated message, in the context of a well-defined artificial (as opposed to natural) language, it can be considered as consisting of a string or sequence of components. Each of these is nothing but a character substring which is properly delimited and in the proper sequential location of the component string.

Thus, given the following conditions:

1. The uniformizer has received any message, i.e., component string, consisting of m components or parameters, for $m \geq 0$, and

2. A reference point j has been established, either explicitly or implicitly during interprocessor interaction, to focus on the jth component, for $1 \leq j \leq m$,

the system must be able to effectively address that component for purposes of utilizing the character string matching capability (described in the previous subsection) and/or the message (re)construction capability (described in the next subsection). The latter alternative really justifies this "component selection" as a separately described capability (distinct from character string matching) in that at times the uniformizer need not match the jth component with anything but rather need simply locate and replace it with a different component or parameter.

Example: In response to a user command to search for

some specified logical combination of subject terms, the retrieval system assigns a sequential number N to that search command for purposes of later reference by the user (see also Section 1.2.1) and then precedes the search results with

SEARCH STATEMENT N:...

Because this is done consistently and because the uniformizer may have to assign a different sequential number to that same search statement (see Section 3.3.4 below), the number N can simply be viewed as the third component (with j=3), assuming components are separated by spaces. That component or parameter need not be matched against some character string (unlike Section 3.3.1) but instead must only be located and replaced.

3.3.3 (Re)construction of Messages

Both above-defined capabilities are in effect prerequisite to implementing the more substantive capability which can render different interaction languages to appear uniform from the standpoint of the user. That capability is the construction or reconstruction of messages received by the uniformizer during one or more interchanges with either the user or the remote system.

Thus, given the following conditions:

1. The uniformizer has received one out of the known set of k (types of) messages available to the user in the common language, or one out of the known set of r (types of) responses reproducible by the jth remote system, and

2. That message is identified to be a particular type, by means of the previously defined capabilities or through sensitivity to the context of known (or expected) interchange sequences,

the system must be able to transform that message, or perhaps utilize its substance, towards the ultimate creation of an appropriate command to the remote system or an appropriate response to the user. Such transformation can range from a simple substitution of a component or parameter (as suggested in the previous subsection), to the re-arrangement of components, to the total replacement of the message by an alternative form. To achieve that end,

the transformation may be

a. One-to-one, in that a single message from the user (or remote system) is transformed into a single message to the remote system (or user), or

b. One-to-many, in that a single message from the user (or remote system) is broken down into a succession of two or more messages to the remote system (or user), or

c. Many-to-one, in that two or more messages from the user (or remote system) are compiled into a single message transmitted to the remote system (or user).

Each choice of transformation will depend on one or more of: what is required by the remote system involved, what is deemed to be desirable in behalf of the user, and the nature of the common command language that is in fact designed and its ability to accommodate both user and system needs.

Example: A user command to search for a logical combination of subject terms must be transformed one-to-one, into the

FIND <arguments> [CR]

command for one retrival system (ORBIT). On the other hand, for a different system (DIALOG), it initiates a one-to-many transformation involving the following three (or more) commands

SELECT <search term> [CR]
SELECT <search term> [CR]
COMBINE <arguments> [CR]

where <arguments> consists of a logical combination of numeric identifiers assigned by the system to each SELECT command respectively. (See also Section 1.2.1).

3.3.4 Mapping of Message Sequences

The three above-described capabilities focus primarily on the individual message-specific level, whether that message is transmitted by the user or the system. It furthermore becomes necessary for the uniformizer to acquire, structure and utilize certain kinds of "knowledge" which are session-(or subsession-) oriented, ranging across the successive sequence of messages emanating either from the user or the system. This is particularly true when a one-to-many transformation (see above) is involved.

When the system creates an explicit interdependence between successive commands c and c+1 issued by the uniformizer, by assigning a sequence of numbers to them, then the correspondence between each single user-input command and the multiple uniformizer-transmitted commands must be developed and maintained for subsequent reference.

Example: Assuming that either one-to-one or one-to-many transformations are involved and given the sequencing of search statements mentioned in both of the previous sections, the uniformizer may have to maintain the following kind of correspondence between the user-understood and the system-required sequences of numbers:

USER	SYSTEM
1	1,2
2	3
3	4,5,6
4	7,8
5	9

Thus, whenever the user makes reference to one of the search statements already previously entered and identified (e.g. number 4), the uniformizer must of course translate that statement number into the system-understood number(s) (e.g. numbers 7 and 8 above).

3.3.5 Handling of Contingencies

All of the previously described capabilities require backup to insure failsafe intermediary operation in the user-system communication channel. For example, a character string, message component or entire message may be out-of-place, improperly delimited or unidentifiable. In these and other instances, the intermediary system must either be smart enough to take corrective action in behalf of the transmitting party (user or object system) or otherwise gracefully react with appropriate error messages or diagnostic inquiries designed to overcome or circumvent the problem situation.

4. TESTBED FACILITIES

The functional requirements of the intermediary uniformizer can now be exemplified through a realistic implementation, using hardware and software facilities available at the National Bureau of Standards.

4.1 Network Access Machine

A minicomputer-based system has been developed at the National Bureau of Standards to enable the implementation and testing of the kinds of intermediary functions described in Section 3. This Network Access Machine (or NAM) [Rosenthal, 1976 a and b] can provide a user with automatic access to any of the large assortment of remotely accessible, interactive computer services. It does this, in general, by allowing a user terminal to connect (or dial in) to it, then establishing the appropriate communications connection to the remote host requested by the user, and then carrying out the previously described intermediary action roles until it is time to sever the user-system connection.

More specifically, the NAM is actually a non-privileged user program which runs under the UNIX operating system. It can thereby take advantage of UNIX capabilities and, as a result, it reflects some of these in terms of what it can accomplish and how. The intention here is not to describe UNIX or the NAM in their full technical detail. Such description has already been produced by Rosenthal and Lucas [1978]. Instead, this report only highlights those special characteristics, such as in communication modes and language constructs, which can support the "uniformizer" requirements outlined in Section 3. This section should therefore provide a general appreciation of what is logically possible.

4.2 Communication Modes

A user wishing to access a remote system via the NAM must establish a hardwire or dial-up connection to the UNIX machine and then carry out the required login procedure (including speed detection for the user's terminal). Subsequently, the user has the repertoire of NAM/UNIX commands available to connect and then interact with the remote system in various direct or indirect (i.e., intermediary assisted) ways. The command types and constructs will be characterized below.

Before doing so, it is important to recognize that the user has a choice between two basic modes of communication:

1. Priority or default communication with the NAM, while direct interaction with the remote host is optional.

2. Direct store-and-forward communication with the remote host, while interaction with the NAM is optional.

A particular command (.Remote (Specified Host)) invokes the store-and-forward mode; a special character (control back-arrow) returns communication to the NAM mode.

4.3 NAM Command Repertoire

Three basic types of commands are available to the NAM user: regular UNIX commands, macro directives (or "quick verbs") and macro names. These are supplemented by a set of string manipulation capabilities. All will be defined and discussed more fully in following subsections. Collectively, they represent the repertoire of commands available to the user (or representative of users) for purposes of constructing the intermediary libraries of software procedures or constructs which can serve the objective of achieving uniformity in interaction language. That is to say, the intermediary system can thereby be equipped with the required transmission, translation, and analysis capabilities which will support the appearance of language uniformity to individual and/or groups of users. In particular, the syntax and semantics of the uniform language are based on the defined macros and the specific arguments which those macros accept.

The three types of commands are distinguished by means of differing preceding special characters (or lack thereof) on input to the NAM.

4.3.1 UNIX Commands

The various UNIX commands and subsystems are utilized in the creation and maintenance of macros. All UNIX commands as well as the UNIX editor are available to the NAM user. A user-input command is assumed to be a UNIX command if no special prefix character is typed. This default treatment is easily changed to apply to the macro commands instead.

Example UNIX commands are:

Filename	Function
pr	To format output for printout
time	To determine time
who	To determine who is on system

4.3.2 Macro Directives

The second type of command is the macro directive which involves a "quick verb" or action request. It causes the NAM to immediately carry out the specified action relative to such general functions as:

1. remote system connection/disconnection

2. data transmission, collection, and output

3. data analysis and related conditions

A macro directive is distinguished from a UNIX command and a macro name command by preceding it with a period.

Examples of macro directives, most of them indicative of above-indicated function categories, are:

Form	Specific Function
.connect (argument)	to establish hardware connection to specified remote system attached to NBS patch panel
.dial (argument)	to establish dial-up connection to specified remote system using automatic calling unit
.msg (argument)	to transmit specified string to user terminal
.send (argument)	to transmit specified character string to remote system; used in conjunction with other directives, e.g., .match and .term
.match (argument)	to define the conditions (i.e., character strings) upon which termination of remote system response is accepted
.term (argument)	to create the conditions (i.e., character strings) which define the termination of remote system response

.newdir (argument) switch to specified working directory

The previously mentioned (in Section 4.2) .remote command is another example macro directive. All of these commands can either be issued directly from the user terminal or they can be incorporated in macro definitions and then executed upon calls to such macros. To further enhance these capabilities, various UNIX commands can be included within macros as if they were macro directives.

Table 1 lists an example macro which is used in this section to illustrate a number of the NAM facilities being discussed. In particular, it includes several of the macro directives, e.g., .msg <argument> and .send <argument>, identified above. Appendix A gives further examples of macro directives within macro definitions.

4.3.3 Conditional Constructs

The commands described thus far can be viewed as building blocks for creating useful macros which can then be saved and, upon being called, expanded and executed by the intermediary processor. But to enable effective macro creation, conditional constructs are required.

Thus a special set of conditional directives, distinguished by being preceded with an asterisk, has been implemented. This includes the well-known IF-THEN-ELSE construct with the following format:

```
        *if <expression>
        ..various directives...
    [*else
        ..various directives...]
        *end
```

where the optional ELSE component is bracketed. Three of these are used in the macro displayed in Table 1.

Also the WHILE-DO construct is available:

```
        *while <expression>
        ..various directives...
        *end
```

See DIALOG's FIND macro in Appendix A for an example. Integer-valued expressions are tested and results determine which directives are executed. A "switch to various cases," which is similar to the directed GO TO, is available in the following form:

```
        *switch <expression>
        ..various directives...
    *case <expression>
        ..various directives...
        [etc.]
```

```
.string s
.int i

.term "t60 ! '? '"
.match "'?'"

*if ( userline ? ".* ( 'database names' | 'file names' ) " )
        .send "?files[CR]"

        .msg response @ ".* '?files' {.*} '?'"
        .set scratch to response

        .exit
    *end

*if ( userline ? ".* ( 'terms' | 'words' )" )
        .set s to userline @ ".* 'related to '{.*}$"
        .send "expand " . s . "[CR]"

        .msg response @ ".* '[CR]' {.*} '-more-' "

        .exit
    *end

*if ( userline ? ".* 'citation'" )
        .set i to userline @ ".* {/0-9//0-9/*} ' citation'"
        .send "type " . flag0 . "/2/1-" . i . "[CR]"

        .msg response @ ".* '[CR]' {.*} '? ' "

        .exit
    *end

.msg "display <object>"
.msg "<object>    := file names | database names | terms related to |"
.msg "               words related to | <number of> citations"
.msg "<number of> := a number"
```

Table 1. The DISPLAY macro (for DIALOG System).

```
[*case <expression>
     ..various directives...]
     *end
```

Here two or more cases can be specified, and control will be transferred to that subset of directives for which the expression matches the one in the switch statement.

4.3.4 String Manipulation

To satisfy the functional requirements outlined in Section 3.3. a set of SNOBOL-like character string manipulation capabilities have been implemented [Rosenthal and Lucas, 1978]. Following is a brief description of each. Specific examples are displayed in Table 1 and among the macros listed in Appendix A.

Firstly, both string and numeric variables can be declared by means of the directives

```
.STRING X and .INT I
```

respectively (see top of Table 1). Then any desirable character string or pattern can be constructed and assigned to a string variable by means of the .set directive, e.g.

```
.SET X to "ABC"
```

This is done for string variable s in Table 1. It then enables easy referencing of any character string, including the entire messages received from or sent to either the user terminal or the remote system.

To perform the actual character string manipulations, three binary operations are defined: concatenation, string matching and substring extraction. They are symbolized by the characters ., ?, and @ respectively. These operators are interpreted, along with the various arithmetic operators (e.g. -, +), whenever the available expression evaluator is invoked. The latter applies to the directives .msg, .send., and .set, and the conditional constructs *if, *while, *switch, and *case.

The concatenation operation (.) is simply demonstrated as follows:

```
.STRING X Y
.SET X to "OUT"
.SET Y to "PUT"
.MSG X . Y
```

generates and transmits the following character string to the user terminal:

```
OUTPUT
```

In the middle of Table 1, this capability is illustrated by

```
.Send "expand" .s. "[CR]."
```

The string matching operation (?) involves the matching of a specified pattern against a string, using the expression format

```
<string> ? <pattern>
```

If a match is determined, the expression value is set to 1 (i.e., TRUE) and, if not, to 0 (i.e., FALSE). Notice that this matching operation is used in each logical expression of the three IF-THEN constructs in Table 1. In each case, the pattern following the question mark (?) is sought in the user-input message called userline.

The third kind of operation, namely substring extraction (@), subsumes the string matching operation and also produces a new string based on extractions from the string in accordance with the elements of the pattern. The expression format is similar to that for string matching:

```
<string> @ <pattern>
```

Extraction results in the return of a (sub)string of characters extracted from the left-side string in the expression. This is essentially demonstrated by the following sample pattern:

```
"({/0-9/}1.)*$"
```

which extracts all numeric characters from the specified string. For example, the above pattern would cause the extraction of numeric characters 1214 from string a1b2c1d4. Another example is given in the third IF-THEN clause of Table 1 where variable i is set to the two-digit number extracted from the string called userline.

The above indicated patterns, for either string matching or extraction, must be carefully constructed into a sequence of elements consisting of some combination of the following symbols and corresponding matching objectives:

.	Any single character
/x-y/	Any character in range x to y, where x and y are single characters
' '	The sequence of characters specified between the single quotes

$	The null character at the end of the string
*	Zero or more occurrences of preceding item
|	The OR operator separating alternative items

Given these operators and their interpretations, following is an example set of string-matching expressions each of which would be evaluated to be TRUE:

```
"F" ? "/C-G/"
"Go" ? "/A-H/  /i-p/"
"my" ? "./x-z/"
"ijk" ? "'ij'"
"mn" ? "'mn'$"
"abcde" ? ".*('fg'l'cd')"
```

These examples are only indicative of available capabilities. They nevertheless suggest how very complex patterns can be constructed and utlized in string-matching and extraction expressions.

4.3.5 Macros

The UNIX commands, macro directives, conditional constructs, and string manipulation capabilities can now be employed collectively to create macros. By means of the UNIX editor, the appropriate commands can be selected, sequenced, and stored in a file. The file names themselves then become constituents of the kind of "uniform" command language which is the primary object of this report. That is to say, a user-input command, which is actually a file name preceded by the special up-arrow character (distinguishing macros from UNIX commands and macro directives), invokes the macro expander and causes the various directives contained in the macro to be executed relative to the particular remote system and interactive user.

Thus, for example, if the user is already logged into the Lockheed DIALOG system, and wishes to use a command of the general form

```
DISPLAY <argument(s)>[CR]
```

where argument(s) may be a list of available filenames, a

list of search terms related to a specified term, or a listing of N citations responsive to a previous search command, the macro shown in Table 1 would be invoked and executed.

Other "real" object system commands can be implemented likewise in terms of macros cntrolled by the NAM. They are included in Appendix A, for two of the systems (ORBIT and DIALOG) involved in our application.

4.4 Other Features

Among the useful features of the macro expander are its recursiveness and nesting capabilities. That is to say, it can call upon itself repeatedly as well as call on other macros (i.e., file names) for their subsequent expansion within the context of the outer (or calling) macro.

Finally, the testbed faclities exhibit certain user-oriented features. In particular, the user who wishes to call upon the various command alternatives previously described can take advantage of the TENEX-like capability for command completion. Thus, a command input in te following order:

Special char (if any), unique char string, Esc/Altmode

will be automatically completed by the system and executed as appropriate.

5. LANGUAGE DESIGN

In order to utilize the framework of logical capabilities and implemented facilities (described in Sections 3 and 4 respectively) for providing uniformity to user-computer interaction languages, a common command language was required. This language had to be tailored, of course, to the selected application, namely bibliographic retrieval.

5.1 Criteria for Command Language Design

Given a number of different systems designed to carry out essentially the same kinds of application-specific functions, how does one design a common command language? Several approaches suggest themselves, including trying to identify from among the different command languages which have somehow evolved those commands which seem to have gained majority acceptance.

Another approach could involve selecting a sample of totally inexperienced computer users, setting them in front of an on-line terminal to a retrieval system and asking them how they would command the computer to retrieve certain bibliographic citations for them. By closely observing, recording, and analyzing their responses, a more "natural" and less biased set of commands could perhaps be constructed.

Instead, the approach adopted was to use personal experience and intuition in establishing a set of design criteria and then proceeding to develop the command language accordingly. The resulting language is therefore not proposed as a "standard" as such, but rather as a flexible model of what such a language might conceivably resemble.

The design criteria were:

1. The command structure should adhere basically to the format of the "action primitive" [Treu, 1975], which consists of
   a. an action verb
   b. qualifier(s) (if any) of that action
   c. an object of the action
   d. qualifier(s) (if any) of the object

   Whenever possible and reasonable, these should be maintained in the indicated order.

2. The action verbs, objects, and qualifiers should be selected to be both meaningful (relative to retrieval system functions) as well as versatile (e.g., in associating a verb with several different objects, or vice versa). See criteron (8).

3. In view of the above, the commands need not be consistent with some of the commonly accepted forms which have evolved over the years but which were normally not selected in a deliberative, systematic way. However, when they happen to fit into the above-suggested format, their inclusion is welcome.

4. The action verbs, objects, and qualifiers should also be selected to be mnemonically distinguishable, rendering each verb unique in its first several characters, and likewise for objects and certain qualifiers. See also criteria (9) and (10).

5. Because some of the different system functions cannot be reconciled via a common language (at this time), the user should be able to bypass or supplement the common commands. To do so, the user should merely have to precede any "real" command by some special character. Thus, the common command

language represents the normal or default mode, but the user can switch to the object system's language directly upon explicit request.

6. To provide a flexible model language, easy modification of or substitution for any given action verb, object or other components, within the established structure of the language, should be possible.

7. Two or more alternative versions of action verbs, objects, and qualifiers should be implemented from the outset, as long as they are selected to avoid confusion with respect to criterion (4).

8. Identical action verbs should be used for different functions when the involved actions are most likely to be conceived by the user as very similar if not exactly the same. An example is the request to "display" some kind of information. The functions can then be distinguished by means of the specified differing action objects.

9. The user should be able to specify only the first several characters and then cause the intermediary system to supply the remaining characters of each known command component automatically, after simply pressing the excape key.

10. In conjunction with the TENEX-like behavior suggested by criterion (9), the language should also attempt to accommodate, as possible, differing requirements of experienced and inexperienced users. A simple first step is to enable the experienced user to inhibit completion of command components at the user terminal, by typing a special character (instead of the escape key).

5.2 Command Language Subset

The subset of functions selected for command language implementation is listed in Table 2. A compact table of that form is envisioned to be the minimum form of user aid which might be posted near the user terminal. The user is, of course, expected to be able to relate to the types of functions applicable to bibliographic retrieval systems and, upon selection of the system of interest and after commanding the NAM to establish connection, e.g., with

CONNECT TO SYSTEM ORBIT (CR)

the user then proceeds to employ any other available action verb/object/ qualifier commands. For examples, to select from among the repertoire of data bases accessible in a

TABLE 2. A COMMON COMMAND LANGUAGE SUBSET FOR BIBLIOGRAPHIC INFORMATION RETRIEVAL USABLE (VIA THE NAM) ON THE ORBIT, MEDLINE, DIALOG, RECON AND BASIS SYSTEMS

TO ACCOMPLISH	THE USER TYPES	WITH PARAMETER(S) OR CONDITION(S)
CONNECTION WITH AND LOGGING INTO THE SELECTED SYSTEM	CONNECT TO SYSTEM <NAME> (CR)	WHERE <NAME> IS ONE OF ABOVE SYSTEM NAMES
IDENTIFICATION OF AVAILABLE FILE OR DATABASE NAMES	DISPLAY FILE NAMES (CR)	
ACCESS TO ONE OF THE AVAILABLE FILES OR DATABASES	ACCESS FILE <FILE NAME> (CR)	
IDENTIFICATION OF RELATED SEARCH TERMS IN APPLICABLE VOCABULARY	DISPLAY TERMS RELATED TO <TERM> (CR)	WHERE <TERM> IS A SELECTED SEARCH TERM
DETERMINATION OF NUMBER OF AVAILABLE RESPONSES OR CITATIONS	FIND <STRING> (CR)	WHERE <STRING> IS SEQUENCE OF TERMS LOGICALLY CONNECTED BY AND, OR, AND NOT
OUTPUT OF SPECIFIED NUMBER OF CURRENT RESPONSES OR CITATIONS	DISPLAY <N> CITATIONS (CR)	WHERE <N> IS DESIRED, NUMBER STARTING WITH FIRST
EXPLANTION OF ANY AVAILABLE (COMMON) COMMAND	EXPLAIN <COMMAND NAME> (CR)	
ASSISTANCE WITH RESPECT TO (COMMON) COMMAND LANGUAGE	HELP (CR)	
LOGOUT AND DISCONNECTION FROM SYSTEM/NETWORK	STOP SESSION (CR)	
EDITING: ERASE LAST CHARACTER ERASE CURRENT COMMAND	 ^H ^U	
FACILITATED COMMAND INPUT WITH AUTOMATIC COMMAND COMPLETION	THE FIRST 3 CHARACTERS OF COMMAND SUBSTRING, IDENTIFIED IN BOLD FACE ABOVE, FOLLOWED BY (ESC) E.G. DIS (ESC) <N> CIT (ESC) (CR)	APPLICABLE TO ALL COMMANDS AS INDICATED ABOVE

given system,

ACCESS FILE NTIS (CR)

and then to actually carry out a search,

FIND NETWORK AND GRAPHICS (CR)

To output a specified number of responses resulting from that search, the user can command

DISPLAY 15 CITATIONS (CR)

and when some help or explanation is desired, corresponding commands are available. Then, after perhaps having utilized various other action commands identified in Table 2, the user can

STOP SESSION (CR)

and thereby cause the NAM to carry out the necessary logout and disconnect procedure.

Several of the details associated with Table 2 warrant elaboration:

1. Automatic command completion applies to both of the following command components:

   a. Action verbs (including alternative versions) collectively, and
   b. Within each function, hierarchically under the set of action verbs, for the set of objects.

   Thus, for example, to carry out display of available filenames, the user would actually only need to type:

   DIS(Esc) FIL(Esc) (CR)

   while the NAM would supply the missing character strings automatically.

2. Qualifiers may be required or optional, depending on command type. The NAM is easily prepared to search for one or more successive qualifiers succeeding each action verb and each object specification. In the interest of more natural interaction, some reasonable exceptions (e.g., with the above-stated DISPLAY command involving the numeric qualifier "15") are possible.

3. Certain qualifiers must get special separate, context- specific attention, e.g., the BOOLEAN operators (AND, OR, NOT) which are required in the term entry and logic creation of the search request.

4. The two input editing functions are not as yet handled in a more user-oriented manner due to a conflict between generally acknowledged, appropriate characters and what the NAM happens to do with them. Perhaps something like Control E (for last character ERASE) and Control I (for IGNORE or erase current line) will be possible.

5.5 Implemented Macros

Utilizing the capabilities of the testbed facilities described in Section 4, and given the detailed descriptions of command semantics and syntax pertaining to the five available retrieval systems, the common command language subset was appropriately specified and then was implemented by Rosenthal and Lucas [1978]. The resulting macros are displayed in Appendix A for two of the systems used.

5.6 Demonstration and Testing

The common command language was then informally tested. For sake of consistency, a predefined scenario of commands, involving most of the implemented functions (See Table 2) at least once, was employed in successively accessing and using two of the five systems. The interactive sessions, as seen by the user as well as by the NAM, were recorded in detail. A sample of each is listed in APPENDIX B.

6. CONCLUSIONS

This report has presented the framework for the motivation, implementation and testing of an intermediary, "uniformizing" processor of user-computer interaction languages. In particular, the NBS testbed facilities, based on the NAM, have been described with primary regard to the conceptual and logical capabilities they must possess in order to effectively serve the needs of on-line users of bibliographic information retrieval systems.

As a consequence of the informal testing and several formal demonstrations of the described testbed facilities, the following conclusions are indicated:

1. For an application of the type employed (i.e., bibliographic retrieval), involving considerable similarities or consistencies among the functions carried out by different system implementations, the uniformizer role (of the NAM) represents a "virtual" form of achieving a common or standardized interaction language. It may be a precursor to "real" standardization in the future.

2. The command language subset presented gives a reasonably effective, first approximation of what might become a more user-oriented common (bibliographic retrieval) language. However, to arrive at a more complete and operationally usable language, more detailed consideration and implementation of necessary or desirable functions and options must be carried out. This should be done in consultation with operational staff persons

or information specialists.

3. For purposes of producing demonstrable results within a reasonable time period, various language options were implemented in minimal form or by default. Much more elegant and powerful features can be incorporated (with present facilities) and, furthermore, could be incorporated (with extended facilities) through committed efforts to further NAM development. Example areas of attention would be user/system profile creation and manipulation as well as "adaptability" to user and system needs.

4. The framework and facilities described also show potentials for applicability to a number of other areas (or systems) involving user-computer interaction. A good example would be the area of text editing. Numerous different text editors are available and user access to them could be "uniformized" by identifying commonalities in function and designing and implementing a common text editing command language. An NBS effort is currently being initiated in this direction.

REFERENCES

Abrams, M. D. and Cotton, I. W. [1975] The Service Concept Applied to Computer Networks, NBS Technical Note 880, August 1975; 34pp.

Abrams, M. D. and Treu, S. [1977] "A Methodology for Interactive Computer Service Measurement," Communications of the ACM, Vol. 20, No. 12, December 1977; pp. 936-944.

Anderson, R. H. and Gillogly, J. J. [1976] "The RAND Intelligent Terminal Agent (RITA) as a Network Access Aid," Proceedings of 1976 NCC, pp. 501-509.

Bennett, J. L. [1972] "The User Interface in Interactive Systems," Annual Review of Information Science and Technology, (Ed. by C. Cusdra), Vol. 7, pp. 159-196.

Berk, T. (Ed.) [1976] Proceedings of ACM Symposium on Graphic Languages, April 26-27, 1976, Miami, Florida, sponsored by ACM SIGGRAPH, ACM SIGPLAN and Florida International U., 128 pp.

Blanc, R. P. [1974] "Assisting Network Users with a Network Access Machine," 1974 Annual Conference of the ACM, pp. 74-84.

Chapanis, A. [1975] "Interactive Human Communication," Scientific American, Vol. 232, No. 3, March 1975, pp. 36-42.

Cheriton, D. R. [1976] "Man-Machine Design for Timesharing Systems," 1976 Annual Conference of the ACM, pp. 362-366.

Enslow, P. H. [1975] "Operating System Command Languages, A Brief History of Their Study," Command Languages, C. Unger, Ed., North-Holland/American Elsevier, pp. 5-24.

Fitzgerald, M. L. [1978] Common Command Language for File Manipulation and Network Job Execution, NBS Special Publication Series 500-37.

Foley, J. D. and Wallace, V. L. "The Art of Natural Graphic Man-Machine Conversation," IEEE Proceedings, Special Issue on Computer Graphics, April 1974.

Mamrak, S. A. and Amer, P. D. [1979] A Methodology for the Selection of Interactive Computer Services, NBS Special Publication 500-44, January 1979.

Marcus, R. S. and Reintjes, J. F. [1976] The Networking of Interactive Bibliographic Retrieval Systems, M.I.T., Report ESL-R-656, NSF Grant SIS74-18165, 164 pp.

Martin, T. H. [1974] A Feature Analysis of Interactive Retrieval Systems, Stanford University, Report SU-COMM-ICR-74-1, NSF Grant GN 36160, 100 pp.

Muchnick, S. S. [1976] "The Command Interpreter and Command Language Design of the COM-SHARE COMMANDER IT System," 1976 Annual Conference of the ACM, pp. 367-372.

Pyke, T. N. Jr. [1973] "Some Technical Considerations for Improved Service to Computer Users," COMCON 73.

Rosenthal, R. [1976 a] A Review of Network Access Techniques with a Case Study: The Network Access Machine, NBS Technical Note 917, July 1976, 29 pp.

Rosenthal, R. [1976 b] "Network Access Techniques - A Review," Proceedings of 1976 NCC, pp. 495-500.

Rosenthal, R. and Lucas, B. D. [1978] The Design and Implementation of the National Bureau of Standards' Network Access Machine (NAM), NBS Special Publication 500-35, June 1978.

Rouse, W. B. [1975] "Design of Man-Computer Interfaces for On-Line Interactive Systems," IEEE Proceedings, Vol. 63, No. 6, June 1975.

Treu, S. [1975] "Interactive Command Language Design Based on Required Mental Work," International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 135-149.

Treu, S. (Ed.) [1977 a] User-Oriented Design of Interactive Graphics Systems, Based on ACM/SIGGRAPH Workshop, October 14-15, 1976, Pittsburgh, Pennsylvania, 162 pp.

Treu, S. [1977 b] "A Framework of Characteristics Applicable to Graphical User-Computer Interaction," User-Oriented Design of Interative Graphics Systems, ACM.

APPENDIX A: Command Macros

The macros, which were designed and implemented for purposes of uniformly performing the functions represented by the common command language subset on each of the sample bibliographic retrieval systems, are listed on the following pages for the systems named ORBIT and DIALOG. The reader may find the indicated hierarchy among the macros useful. Each macro name is also identified by page number (referring to this Appendix) to easily locate its listing:

CONNECT (page A2)

(if ORBIT is selected: page A3)

DISPLAY (page A4)
ACCESS (page A5)
FIND (page A6)
HELP (page A7)
STOP (page A8)

(if DIALOG is selected: page A9)

DISPLAY (page A10)
ACCESS (page A11)
FIND (page A12)
HELP (page A13)
STOP (page A14)

It is interesting to observe, for example, how much more work has to be done by the common language's FIND command when employed for the DIALOG system (see page A12), with its SELECT ann COMBINE scenario, as opposed to the ORBIT system (see page A6), which already has a command quite consistent with the FIND.

NOTE: See Appendix B for transcripts of example user sessions.

CONNECT MACRO

```
.string sl name

.set sl to " .* { 'medline' | 'orbit' | 'recon' | 'dialog' |
basis' } "
.set name to userline @ sl

*if ( name = "" )
          .msg "connect to system <name>"
          .msg "<name> := medline | orbit | recon | dialog |
          basis"
          .exit
     *end

.newdir $name.
connect
```

```
.set connectionmade = 0
.connect tymnet orbit -p

*if ~ connectionmade
          .msg "sorry, please try again"
          .newdir ..
          .exit
     *end

.transcript-on
.term "t30 ! 'identifier" ! ':' ! ';' ! '|'"
.match "'.'"
.send ""
*if ( response ? ".*'|'" )
          .msg "remote connection error [CR][LF]"
          .msg "no response, please try again"
          .disconnect
          .newdir ..
          .exit
     *end

*if ( response ? ".*'identifier'" )
          .send "e"
     *end

*if ( response ? ".* ( 'name' | 'log in' ) " )
          .send "___[CR]"
     *else
          .msg "remote connection sequence error"
          .msg "please try again"
          .disconnect
          .newdir ..
          .exit
     *end

.send "orbit [CR]"
.term "t60 ! 'USER:'"
.send "/login ________[CR]"
*if ( response ? ".* 'USER:'" )
          .send "n[CR]"
          .msg "Login successful [CR][LF]"
          .exit
     *end

.msg "Problem logging into ORBIT [CR][LF]"
.msg response
.disconnect
.newdir ..
```

```
.string s
.int i

.term "t60 ! 'user:[CR][LF]' ! 'USER:[CR][LF]'"
.match "':'"

*if ( userline ? ".* ( 'database names' | 'file names' ) " )
          .send "\"files\"[CR]"

          .msg response @ ".* 'PROG:' {.*} 'SS ' /0-9/ "

          .exit
     *end

*if ( userline ? ".* ( 'terms' | 'words' )" )
          .set s to userline @ ".* 'related to '{.*} '[LF]'"

          .send "\"nbr " . s . "\" [CR]"

          .msg response @ ".* 'PROG:' {.*} 'UP N OR DOWN N?'
          "

          .send "0[CR]"
          .exit
     *end

*if ( userline ? ".* 'citation'" )
          .set i to userline @ ".* {/-9//0-9/*} ' citation'
          "
          .send "\"prt " . i . "\"[CR]"

          .msg response @ ".* 'PROG:' {.*} 'SS ' /0-9/ "

          .exit
     *end
.msg "display <object>"
.msg "<object>    := file names | database names | terms
related to |"
.msg "               words related to | <number of>
citations"
.msg "<number of>:= a number"
```

```
.term "t60 ! 'user:[CR][LF]' ! 'USER:[CR][LF]'"
.match "':'"
.send "\"file "$2\". "[CR]"p
.msg response @ ".* 'PROG:' {.*} 'SS ' /0-9/ "
```

```
.string s ssnopat ssno hitspat hits

.set s to userline @ "'find'{.*}$"
.set hitspat to    ".* 'PSTG'.* '(' {/0-9/.*} ')'"
.set ssnopat to    ".* 'SS'.* 'SS ' {/0-9/.*} ' '"
.term "t30 ! 'user:\[cr lf]' ! 'USER:\[cr lf]"
.match "'.'"
.send s . "[CR]"

.set hits to response @ hitspat

*if ( response ? ".* ( 'NONE-' | 'NP (' ) " )
          .set hits to "0"
          .set ssnopat to ".* 'SS '{/0-9/.*} ' '"
          *end

.set ssno to response @ ssnopat

.msg "[CR][LF]" . hits . " hits on " . s
.msg "Enter next search (# " . ssno . " )"
```

```
.msg "you have the following nam commands:"
.msg " "
.msg "connect to system <name>"
.msg "<name> := medline | orbit | recon | dialog | basis"
.msg " "
.msg "display <object>"
.msg "        <object>    := file names | database names |
terms related to |"
.msg "                       words related to | <number of>
citations"
.msg "        <number of> := a number"
.msg " "
.msg "        examples:  display words related to oil
embargo"
.msg "                   display 10 citations"
.msg " "
.msg "access file <name>"
.msg " "
.msg "find <search list>"
.msg "        example:  find wheat and russia and not china"
.msg " "
.msg "explain <command>"
.msg " "
.msg "stop"
.msg " "
.msg "-- editing --"
.msg "   erase character is control h (backspace)"
.msg "   erase line is control u (nak)"
.msg " "
.msg "_ Direct Host Communication --"
.msg ".remote              to communicate directly with host"
.msg "control back arrow  to return back to the NAM"
```

```
.term "t30 ! 'user:\[cr lf]' ! 'USER:\[cr lf]'"
.match "'done?'"

.send "\"stop\"[CR]"

.term "t30 ! 'good-bye' ! 'GOOD-BYE'"
.send "y[CR]"

.msg response @ ".* 'PROG:' {.* 'GOOD-BYE' } .* $"

.wait 3
.disconnect orbit
.newdir ..
```

```
; flag0 is used only by dialog to remember the last search
statement
; see macro find

.set flag0 to 1

.set connectionmade to 0
.connect tymnet dialog -p

*if ~ connectionmade
          .msg "sorry, please try again"
          .newdir ..
          .exit
     *end

.transcription
.term "t30 ! 'identifier' ! ':' ! 'bbbb[CR]' ! BBBB[CR]' !
'?' ! '|'"
.match "'id'"

.send ""
*if ( respnse ? ".*'|'" )
          .msg "remote connection error[CR][LF]"
          .msg "no response, please try again"
          .disconnect
          .newdir ..
          .exit
     *end

*if ( response ? ".*'identifier'" )
          .send "e"
     *end

*if ( response ? ".* ( 'name' | 'log in' ) " )
          .send "___[CR]"
     *else
 .msg "remote connection sequence error"
 .msg "please try again"
 .disconnect
 .newdir ..
 .exit
     *end

.send "_______[CR]"
*if ( response ? ".* ( 'error' | 'trouble' | 'busy' | 'down'
)" )
          .msg response
          .msg "please try again"
          .disconnect
          .newdir ..
```

```
          .exit
     *end

*if ( response ? ".* ( 'PASSWORD' | 'password' | 'PASSWORD'
) " )


          .term "t60 ! '? '"
          .send "________[CR]"

          *if ( response ? ".* '? ' " )
               .msg "Login successful[CR][LF]"
               .msg response @ " {.*} '? ' "
               .exit
          *end
     *end

.msg "Problem logging into DIALOG[CR][LF]"
.msg response
.disconnect
.newdir ..
```

```
.string s
.int i

.term "t60 ! '? '"
.match "'?'"

*if ( userline ? ".* ( 'database names' | 'file names' ) " )
          .send "?files[CR]"

          .msg response @ ".* '?files' {.*} '?'"
          .set scratch to response

          .exit
     *end

*if ( userline ? ".* ( 'terms' | 'words' ) " )
          .set s to userline @ ".* 'related to '{.*}S"
          .send "expand " . s . "[CR]"

          .msg response @ ".* '[CR]' {.*} '-more-' "

          .exit
     *end

*if ( userline ? ".* 'citation'" )
          .set i to userline @ ".* {/0-9//0-9/*} '
          citation'"
          .send "type " . flag0 . "/2/1-" . i . "[CR]"

          .msg response @ ".* '[CR]' {.*} '? ' "

          .exit
     *end

.msg "display <object>"
.msg "<object>    := file names | database names | terms
related to |"
.msg "               words related to | <number of>
citations"
.msg "<number of> := a number"
```

```
.string userfilename systemfilename filenumber scratch1
.string namepat matchpat

.set namepat to "......{.*} '[CR]'"

.term "t60 ! '? '"
.match "'?'"

*if ( ~ ( userline ? "'access file '" ) )
.msg "access file <name>"
.exit
     *end

.set userfilename to userline @ "'access file ' {.*} '[LF]'"

*if ( userfilename = "" )
          .msg "access file <name>"
          .exit
     *end

*if ( userfilename ? "/0-9//0-9/* $" )
          .send "begin " . userfilename . "[CR]"
          .msg response @ ".* '[CR]' {.*} '? '"
          .exit
     *end

*if ( ~ ( scratch ? ".* 'file' " ) )
          .send "?files[CR]"
          .set scratch to response
     *end

.set scratch1 to scratch @ ".* ':[CR][LF]' {.*} '[CR][LF]?'

*while ( scratch1 # "" )

     .set filenumber to scratch1 @ ".* {/0-9//0-9/*} ' '"
     .set systemfilename to scratch1 @ namepat

     .set matchpat to ".*'" . userfilename . "'"
     *if ( systemfilename ? $matchpat. )
          .send "begin " . filenumber . "[CR]"
          .msg response @ ".* '[CR]' {,*} '? '"
          .exit
     *end

     .set scratch1 to scratch1 @ ".* '[CR][LF]' {.*} $"
*end

.msg "file " . userfilename . " not found"
```

```
.term "t40 ! '? '"
.match "'?'"

.string s result key next rest ssno ssnopat hits hitspat

.set s to userline @ "'find' {.*} '[LF]'$"
.set next to "' '* ( {/a-z//a-z/*}(' '|'('|')'|$) |
{'('|')'} ')"
.set rest to "' '* ( (/a-z//a-z/*)(' '|{'('|')'}|$) |
('('|')') ) {.*} $"
.set ssnopat to ".*'[CR]'.*{/0-9//0-9/*}' '"
.set hitspat to ".*'[CR]'.*/0-9/.*' ' {/0-9//0-9/*}' ' "
.set result to ""

*while s # " "
          .set key to s @ next
          .set s to s @ rest
          *switch key
          *case "("
          *case ")"
          *case "or"
          *case "and"
          *case "not"
               .set result to result . " " . key
               .break
          *case ""
               .break
          *default
               .flush
               .send "select " . key . "[CR]"
               .set ssno to response @ ssnopat
               .set result to result . " " . ssno
               .break
          *end
*end
.send "combine" . result . "[CR]"
.msg ( response @ hitspat ) . " hits on " . ( userline @
"'find'{.*}$" )
.msg "Enter next search (# " . ( resonse @ ssnopat ) . ")"

.set flag0 to ( ( response @ ssnopat ) - 1 )
```

```
.msg "you have the following nam commands:"
.msg " "
.msg "connect to system <name>"
.msg "<name> := medline | orbit | recon | dialog | basis"
.msg " "
.msg "display <object>"
.msg "        <object>    := file names | database names |
terms related to |"
.msg "                       words related to | <number of>
citations"
.msg "        <number of> := a number"
.msg " "
.msg "        examples:  display words related to oil
embargo"
.msg "                   display 10 citations"
.msg " "
.msg "access file <name>"
.msg " "
.msg "find <search list>"
.msg          example:  find wheat and russia and not china"
.msg " "
.msg "explain <command>"
.msg " "
.msg "stop"
.msg " "
.msg "-- editing --"
.msg "   erase character is control h (backspace)"
.msg "   erase line is control u (nak)"
.msg " "
.msg "-- Direct Host Communication--"
.msg ".remote              to communicate directly with host"
.msg " control back arrow to return back to the NAM"
```

```
.term "t30 ! 'dropped by host system' ! 'DROPPED BY HOST
SYSTEM'"
.match '"'.'"

.send "logoff[CR]"

.msg response @ ".* 'logoff' {.*} 'tc>' "

.wait 2
.disconnect dialog
.newdir ..
```

APPENDIX B: Example Session Transcripts

1. Following pages display the dichotomized transcript of a NAM- supported session conducted with the ORBIT retrieval system. First is the listing of what the user sees and requests in interacting with the NAM. User-input commands are identified by the preceding colon (:). Secondly, the corresponding transactions between the NAM and the remote system are listed. To facilitate reader comparison of the respective transcripts, for purposes of observing what the NAM reaction is to each command entered by the user, the following succession of "common" commands and their corresponding ORBIT commands can be used as keys:

User Command Scenario	Corresponding ORBIT Command
Connect to system orbit	(dial-up and log-in procedure)
Display file names	"files"
Access file ntis	"file ntis"
Display terms related to networks	"nbr networks"
Find networks and graphics	networks and graphics
Display 2 citations	"prt 2"
Stop session	"stop"

The same types of transcripts for the DIALOG retrieval system are introduced on page B7.

```
A Network Access Machine -- NAM
:connect to system orbit
dialing 8419560
Phone connection to 8419560 established
logical connection orbit established

Login successful

:display file names

YOU MAY ACCESS THE ERIC, CHEMCON, CHEM7071, AGRICOLA, NTIS,
ORBIT, TULSA, INFORM, GEOREF, COMPENDEX, LIBCON/E,
POLLUTION, P/E NEWS, BIOSIS PREVIEWS, MANAGEMENT,
ACCOUNTANTS, PNI, SSIE, CRECORD, PAPERCHEM, CDI, DEMO NTIS,
GRANTS, FSTA, LISA, BIOCODES, BIO6973, CBPI, APILIT, CIS
INDEX, ASI, CIN AND ENERGYLINE DATABASES. YOU ARE NOW
CONNECTED TO THE ORBIT DATABASE.



:access file ntis

ELAPSED TIME ON ORBIT:  0.03 HRS.
YOU ARE NOW CONNECTED TO THE NTIS DATABASE.



:display terms related to networks


POSTINGS        TERM
     1        NETWORKED (IT)
    33        NETWORKING (IT)
  3201        NETWORKS (IT)
     3        NETWORKS/OS (IT)
  1092        *NETWORKS (IT)


:find networks and graphics

168 hits on networks and graphics

Enter next search (# 2)

:display 2 citations

-1-
AN   - AD-A041 798/OSL
TI   - Interactive Computer Graphics:  A Responsive Planning
       and Control Tool for DoD Program Management
TNO  - Research rept.
Au   - Callahan, Joseph E.; Roberson, Carlton F.; Perino,
       George, H. Jr.
CS   - Defense Systems Management Coll Fort Belvoir VA
PD   - Jan 77; 56p
```

IS - U7720
PR - NTIS Prices: PC A04/MF A01

-2-
AN - AD-A040 763/5SL
TI - Dynamic Memory Allocation for a Virtual Memory Computer
TNO - Doctoral thesis
AU - Budzinski, Robert Lucius
OS - Illinois Univ At Urbana-Champaign Coordinated Science Lab
PD - Jan 77; 149p
IS - U7718
PR - NTIS Prices: PC A07/MF A01

:stop session

TERMINAL SESSION FINISHED 11/11/77 8:00 A.M. (PACIFIC TIME)
ELAPSED TIME ON NTIS: 0.07 HRS.
TOTAL ELAPSED TIME: 0.10 HRS.

PLEASE HANG UP YOUR TELEPHONE NOW. GOOD-BYE
connection orbit closed

```
please log in:
error, type user name:
error, type user name:  sdc

password:
p 35
;<-
<-YOU ARE ON LINE L9F

HELLO FROM SDC/ORBIT.  (11/11/77 7:54 A.M. PACIFIC TIME)
YOU ARE NOW CONNECTED TO THE ORBIT DATABASE.

ARE YOU A NEW USER?  IF YES ENTER Y.  IF NO ENTER N OR A
COMMAND.

USER:
n

PROG:
****CANADIAN BUSINESS PERIODICALS INDEX ("FILE CBPI") NOW
LOADED
THRU 1975.  MORE NEXT WEEK.
****

SS 1 /C?
USER:
"files"

PROG:

YOU MAY ACCESS THE ERIC, CHEMCON, CHEM7071, AGRICOLA,  NTIS,
ORBIT,   TULSA,   INFORM,   GEOREF,   COMPENDEX,   LIBCON/E,
POLLUTION,   P/E   NEWS,   BIOSIS   PREVIEWS,   MANAGEMENT,
ACCOUNTANTS,  PNI, SSIE, CRECORD, PAPERCHEM, CDI, DEMO NTIS,
GRANTS, FSTA, LISA, BIOCODES,  BIO6973,  CBPI,  APILIT,  CIS
INDEX,  ASI,  CIN  AND  ENERGYLINE  DATABASES.   YOU ARE NOW
CONNECTED TO THE ORBIT DATABASE.

SS 1 /C?
USER:
"file ntis"

PROG:
ELAPSED TIME ON ORBIT:  0.03 HRS.
YOU ARE NOW CONNECTED TO THE NTIS DATABASE.

SS 1 /C?
USER:
"nbr networks"

PROG:

POSTINGS         TERM
     1         NETWORKED (IT)
    33         NETWORKING (IT)
```

```
  3201          NETWORKS (IT)
     3          NETWORKS/OS (IT)
  1092         *NETWORKS (IT)
UP N OR DOWN N?

USER:
0

PROG:
SS 1 /C?
USER:
 networks and graphics

PROG:
SS 1 PSTG (168)


SS 2 /C?
USER:
"prt 2"

PROG:

-1-
AN    - AD-A041 798/OSL
TI    - Interactive Computer Graphics:  A Responsive Planning
        and Control Tool for DoD Program Management
TNO   - Research rept.
AU    - Callahan, Joseph E.; Roberson, Carlton F.; Perino,
        George H. Jr.
PD    - Jan 77; 56p
IS    - U7720
PR    - NTIS Prices:  PC A04/MF A01

-2-
AN    - AD-A040 763/5SL
TI    - Dynamic Memory Allocation for a Virtual Memory
        Computer
TNO   - Doctoral thesis
AU    - Budzinski, Robert Lucius
OS    - Illinois Univ At Urbana-Champaign Coordinated Science
        Lab
PD    - Jan 77; 149p
IS    - U7718
PR    - NTIS Prices:  PC A07/MF A01

SS 2 /C?
USER:
"stop"

PROG:
DONE? (Y/N)

USER:
Y
```

```
PROG:
TERMINAL SESSION FINISHED 11/11/77 8:00 A.M. (PACIFIC TIME)
ELAPSED TIME ON NTIS:  0.07 HRS.
TOTAL ELAPSED TIME:  0.10 HRS.

PLEASE HANG UP YOUR TELEPHONE NOW.  GOOD-BYE
```

2. Following pages display the pair of transcripts resulting from an actual session with the DIALOG retrieval system. Again, as was suggested in Part 1 of this Appendix, the reader can easily observe the NAM role by looking for the following succession of command pairs respectively.

User Command Scenario	Corresponding DIALOG Command(s)
Connect to system dialog	(Dial-up and log-in procedure)
Display file names	Files
Access file 6	Begin 6
Display terms related to networks	Expand networks
Find networks and graphics	Select networks Select graphics Combine 1 and 2
Display 2 citations	Type 2/2/1-2
Stop session	Logoff

Note that the user command scenario is identical to that used with ORBIT in Part 1, except that "6" was used in place of filename NTIS. However the NAM can instead accept the latter, search the system-provided list of filenames for its occurrence and then use the corresponding number (6, in this case) in the command transmitted to the remote system.

```
A Network Access Machine -- NAM
:connect to system dialog
dialing 5551234
Phone connection to 5551234 established
logical connection dialog established

Login successful

RECONNECT File 7 Fri 11nov77 8:29:30


FILE 19 IS NOT WORKING TODAY...
FILE 48 IS NOW FILE 20 FEDERAL INDEX...
FILE 47 IS NOW FILE 21 FED IND WKLY...
FOR PTS FILES REALIGNMENT SEE ?PTS


:display file names

Accessable files:
 1  ERIC 66-77/OCT
 2  CA CONDENS 70-71
 3  CA CONDENS 72-76
 4  CA CONDENS/CASIA 77- /VOL87(18)
 5  BIOSIS PREVIEWS 72-77/OCT
 6* NTIS 64-77/ISS22
 7  SOCIAL SCISEARCH 72-77/WK38
 8  COMPENDEX 70-77/OCT
*9  AIM/ARM 67-76/DEC
10  AGRICOLA 70-77/OCT
11  PSYCH ABS 67-77/SEPT
12  INSPEC-PHYSICS 69-77/ISS20
13  INSPEC-ELEC & COMPUT 69-77/ISS20
14  ISMEC-MECH ENGR 73-77/ISS16
15  ABI/INFORM 71-77/OCT
16  PTS MARKET ABS 72-77/OCT
17  PTS WEEKLY NOV 10,1977
18  PTS F&S INDEXES 72-77/SEPT
19  CHEM IND NOTES 74-77/ISS44
20  FEDERAL INDEX AUG 77
21  FED IND WKLY NOV 9, 1977
22  EIS PLANTS(TYPES $0.50 EA.) OCT 77
23  CLAIMS/CHEM 50-76/DEC
24  CLAIMS/CHEM 77-77/JULY
25  CLAIMS/CLASS JAN77
26  FOUNDATION DIRECTORY JULY77
27  FOUNDATION GRANTS 73-77/VOL18 ISS6
28  OCEANIC ABS 64-77/SEP OCT
29  MET/GEOASTRO ABS 70-76/DEC
30  CASIA 72-76
31  CHEMNAME(TM) 398,822 SUBSTANCES
32  METADEX 66-77/NOV
33  WORLD ALUMINUM ABS 68-77/OCTOBER
34  SCISEARCH 74-77/WK38
35  COMP DISSERT ABS 1861-1977/OCT
```

```
36   LANGUAGE ABS 73-76/SEP
37   SOCIOLOGICAL ABS 63-77/#3
38   AMERICA:  HIST & LIFE VOL3-14A
39   HISTORICAL ABS VOL19-23B
40   ENVIROLINE 71-77/AUGUST
41   POLLUTION ABS 70-77/MAY
42   PHARM NEWS INDEX 75-77/OCT
43   CA PATENT CONCORDANCE 72-77/JUN
44   AQUATIC SCI ABS JAN-JUL 1975
45   APTIC 66-77/SEPTEMBER
46   NICEM 1977
47   (*offline*)
48   (*offline*)
49   PAIS 76-77/BUL.#28-31,FLI.V7#2
50   CAB ABS 72-77/SEPTEMBER
51   FSTA 1969 - NOV 1977
52   TSCA 1977 CANDIDATE LIST
54   CEC/EXCEP CHILD 66-77/APR
56   ART MODERN /74-76
57   CLAIMS/GEM 71-77/SEPT
59   FROST & SULLIVAN DM2/75-76
60   USDA/CRIS 75-77/SEPT
64   CHILD ABUSE AND NEGLECT
69   ENERGYLINE 71-77/JUNE
71   ONTAP TEST
75   (*offline*)
81   PTS US STAT ABS71-77/AUG
82   PTS US ANNUAL TIME SERIES/OCT 77
83   PTS US REG TIME SERIES/JUN 77
86   PTS FRN STAT ABS 71-77/AUG
87   PTS FRN ANNL TIME SERIES/OCT 77


:access file 6

         11nov77 8:32:31 User 1067
  $3.57  .051 Hrs File 7
  $0.41  Tymnet
  $3.98  Estimated Total Cost
File6*:NTIS 64-77/ISS22
         Set Items Description (+=OR;*=AND;-=NOT)
         --- ----- ----------- -----------------

:display terms related to networks

Ref  Index-term              TYPE ITEMS RT
E1    NETWORK VOICE
        CONFERENCING----------      1
E2    NETWORK VOICE PROTOCOL---     1
E3    NETWORK/440-------------      1
E4    NETWORKED---------------      1
E5    NETWORKING--------------     35
E6   -NETWORKS----------------   5055
E7    NETWORKS ANALYSIS THEORY-     1
E8    NETWORKS SYNTHESIS-------     1
```

```
      PROGRAMMING LANGUAGE---      1
E10  NETWORTH----------------      1
E11  NETWORTH COMPUTER
       PROGRAM---------------      1
E12  NETZ--------------------      1
E13  NETZE-------------------      1
E14  NETZKATEGORIEN----------      1
E15  NETZUMLAUF--------------      1
E16  NETZWERKANALYSEPROGRAMM--     1
E17  NETZWERKEN--------------      1

:find networks and graphics
304 hits on networks and graphics

Enter next search (# 3)

:display 2 citations

2/2/1
N77-28015/4ST    NTIS Prices:  PC A04/MF A01

Gasplot - a Computer Graphics Program That Draws  a  Variety
of Thermophysical Property Charts

National  Aeronautics  and  Space  Administration.    Lewis
Research Center, Cleveland, Ohio.

AUTHOR:  Trivisonno, R. J.; Hendricks, R. C.
D3413B4    Fld:  20M, 9B, 7D, 46, 62B, 99F    STAR1518
June 77    58p
Rept No:  NASA-TN-D-8507, E-8627
Monitor:18

Descriptors:     *Computer     programs,     *Thermophysical
properties, *Computer graphics, Fluids, Gases, Fortran,
Graphs (Charts), Plotting, Univac 1100 series computers

Identifiers:  Fortran  5  programming  language,  *Gasplot
computer program, NTISNASA

2/2/2
AD-908 249/6ST    NTIS Prices:  PC A06/MF A01

Graphite/Epoxy,     Boron-Graphite/Epoxy     Hybrid     and
Boron/Aluminum Design Allowables

Grumman Aerospace Corp Bethpage N Y    (388 847)

AUTHOR:  Cairo, Ronald P.; Torczyner, Robert D.
Technical rept. May 71-June 72
D3372B4    Fld:  11D, 13M, 13E    GRA17722
Dec 72    120p
Contract:  F33615-71-C-1605
Project:  AF-6169CW, AF-698CW
Monitor:  AFML-TR-72-232
```

Distribution limitation now removed.

Descriptors: (*Composite materials, *Box beams), (*Graphite, Composite materials), (*Epoxy resins, Composite materials), (*Boron, Composite materials), (*Aluminum alloys, Composite materials), Laminates, Fibers, Filaments, Reinforcing materials, Mechanical properties, Tensile properties, Compressive properties, Stresses, Modulus of elasticity, thermal expansion, Bolted joints, Bonded joints, Mathematical models, Mathematical prediction, Graphics, Computer programs, Shear stresses

Identifiers: Aluminum alloy 6061, Composite materials, Matrix materials, Metals, Poissons ratio, NTISDODXD

:stop session

llnov77 10:07:50 User 1067
$1.89 0.054 Hrs File6* 2 Descriptors
$0.43 Tymnet
$2.32 Estimated Total Cost

LOGOFF 10:07:53
<-<-<-<-
connection dialog closed

```
please type your terminal identifier
-1011-12--
please log in:  ------

password:
<-<-<-<-tc>    host is online


ENTER YOUR DIALOG PASSWORD
7 Fri 11nov77 8:29:30

FILE 19 IS NOT WORKING TODAY...
FILE 48 IS NOW FILE 20 FEDERAL INDEX...
FILE 47 IS NOW FILE 21 FED IND WKLY...
FOR PTS FILES REALIGNMENT SEE ?PTS
? ?files
Accessable files:
 1   ERIC 66-77/OCT
 2   CA CONDENS 70-71
 3   CA CONDENS 72-76
 4   CA CONDENS/CASIA 77- /VOL87(18)
 5   BIOSIS PREVIEWS 72-77/OCT
 6*  NTIS 64-77/ISS22
 7   SOCIAL SCISEARCH 72-77/WK38
 8   COMPENDEX 70-77/OCT
 9   AIM/ARM 67-76/DEC
10   AGRICOLA 70-77/OCT
11   PSYCH ABS 67-77/SEPT
12    INSPEC-PHYSICS 69-77/ISS20
13   INSPEC-ELEC & COMPUT 69-77/ISS20
14   ISMEC-MECH ENGR 73-77/ISS16
15   ABI/INFORM 71-77/OCT
16   PTS MARKET ABS 72-77/OCT
17   PTS WEEKLY NOV 10,1977
18   PTS F&S INDEXES 72-77/SEPT
19   CHEM IND NOTES 74-77/ISS44
20   FEDERAL INDEX AUG 77
21   FED IND WKLY NOV 9,1977
22   EIS PLANTS(TYPES $0.50 EA.) OCT 77
23   CLAIMS/CHEM 50-76/DEC
24   CLAIMS/CHEM 77-77/JULY
25   CLAIMS/CLASS JAN77
26   FOUNDATION DIRECTORY JULY77
27   FOUNDATION GRANTS 73-77/VOL18 ISS6
28   OCEANIC ABS 64-77/SEP-OCT
29   MET/GEOASTRO ABS 70-76/DEC
30   CASIA 72-76
31   CHEMNAME(TM) 398,822 SUBSTANCES
32   METADEX 66-77/NOV
33   WORLD ALUMINUM ABS 68-77/OCTOBER
34   SCISEARCH 74-77/WK38
35   COMP DISSERT ABS 1861-1977/OCT
36   LANGUAGE ABS 73-76/SEP
37   SOCIOLOGICAL ABS 63-77/#3
38   AMERICA:  HIST & LIFE VOL3-14A
```

```
39  HISTORICAL ABS VOL19-23B
40  ENVIROLINE 71-77/AUGUST
41  POLLUTION ABS 70-77/MAY
42  PHARM NEWS INDEX 75-77/OCT
43  CA PATENT CONCORDANCE 72-77/JUN
44  AQUATIC SCI ABS JAN-JUL 1975
45  APTIC 66-77/SEPTEMBER
46  NICEM 1977
47  (*offline*)
48  (*offline*)
49  PAIS 76-77/BUL.#28-31,FLI.V7#2
50  CAB ABS 72-77/SEPTEMBER
51  FSTA 1969 - NOV 1977
52  TSCA 1977 CANDIDATE LIST
54  CEC/EXCEP CHILD 66-77/APR
56  ART MODERN /74-76
57  CLAIMS/GEM 71-77/SEPT
59  FROST & SULLIVAN DM2/75-76
60  USDA/CRIS 75-77/SEPT
64  CHILD ABUSE AND NEGLECT
69  ENERGYLINE 71-77/JUNE
71  ONTAP TEST
75  (*offline*)
81  PTS US STAT ABS71-77/AUG
82  PTS US ANNUAL TIME SERIES/OCT 77
83  PTS US REG TIME SERIES/ JUN 77
86  PTS FRN STAT ABS 71-77/AUG
87  PTS FRN ANNL TIME SERIES/OCT 77
? begin 6
          11nov77 8:32:31 User 1067
  $3.57   0.051 Hrs File7
  $0.41   Tymnet
  $3.98   Estimated Total Cost
File6*:NTIS 64-77/ISS22
          Set Items Description (+=OR;*=AND;-=NOT)
          --- ----- ----------- ------------------
? expand networks

Ref  Index-term              Type Items RT
E1   NETWORK VOICE
       CONFERENCING----------      1
E2   NETWORK VOICE PROTOCOL---      1
E3   NETWORK/440-------------      1
E4   NETWORKED---------------      1
E5   NETWORKING--------------     35
E6  -NETWORKS----------------   5055
E7   NETWORKS ANALYSIS THEORY-      1
E8   NETWORKS SYNTHESIS-------      1
E9   NETWORKS. FORTRAN 4
       PROGRAMMING LANGUAGE---      1
E10  NETWORTH----------------      1
E11  NETWORTH COMPUTER
       PROGRAM----------------      1
E12  NETZ--------------------      1
E13  NETZE-------------------      1
```

```
E14  NETZKATEGORIEN----------      1
E15  NETZUMLAUF--------------      1
E16  NETZWERKANALYSEPROGRAMM--     1
E17  NETZWERKEN--------------      1
                                -more-

? select networks
          1  5055 NETWORKS
? select graphics
          2  4572 GRAPHICS
? combine 1 and 2
          3  304 1 AND 2

? type 2/2/1-2

2/2/1
N77-28015/4ST    NTIS Prices:  PC A04/MF A01

Gasplot - a Computer Graphics Program That Draws  a  Variety
of Thermophysical Property Charts

National  Aeronautics  and  Space  Administration.    Lewis
Research Center, Cleveland, Ohio.

AUTHOR:  Trivisonno, R. J.; Hendricks, R. C.
D3413B4    Fld:  20M, 9B, 7D, 46, 62B, 99F,    STAR1518
Jun 77    58p
Rept No:  NASA-TN-D-8507, E-8627
Monitor:  18

Descriptors:     *Computer     programs,     *Thermophysical
properties,  *Computer  graphics,  Fluids,  Gases,  Fortran,
Graphs (Charts), Plotting, Univac 1100 series computers

Identifiers:  Fortran  5  programming  language,  *Gasplot
computer program, NTISNASA

2/2/2
AD-908 249/6ST    NTIS Prices:  PC A06/MF A01

Graphite/Epoxy,     Boron-Graphite/Epoxy     Hybrid      and
Boron/Aluminum Design Allowables

Grumman Aerospace Corp Bethpage N Y    (388 847)

AUTHOR:  Cairo, Ronald P.; Torczyner, Robert D.
Technical rept. May 71-Jun 72
D3372B4    Fld:  11D, 13M, 13E    GRA17722
Dec 72    120p
Contract:  F33615-71-C-1605
Project:  AF-6169CW, AF-698CW
Monitor:  AFML-TR-72-232
Distribution limitation now removed

Descriptors:    (*Composite    materials,    *Box    beams),
```

(*Graphite, Composite materials), (*Epoxy resins, Composite materials), (*Boron, Composite materials), (*Aluminum alloys, Composite materials), Laminates, Fibers, Filaments, Reinforcing materials, Mechanical properties, Tensile properties, Compressive properties, Stresses, Modulus of elasticity, Thermal expansion, Bolted joints, Bonded joints Mathematical models, Mathematical prediction, Graphics, Computer programs, shear stresses

Identifiers: Aluminum alloy 6061, Composite materials, Matrix materials, Metals, Poissons ration, NTISDODXD

? logoff
llnov77 10:07:50 User 1067
$1.89 0.054 Hrs File6* 2 Descriptors
$0.43 Tymnet
$2.32 Estimated Total Cost

LOGOFF 10:07:53

<-<-<-<-tc> dropped by host system

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-63	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Computer Science & Technology:
A Testbed for Providing Uniformity to User-Computer Interaction Languages

5. Publication Date: August 1980

6. Performing Organization Code

7. AUTHOR(S)

Siegfried Treu

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, DC 20234

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Same as above.

13. Type of Report & Period Covered: Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 80-600107

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The differing user-computer interaction languages, implemented for conducting the same applications-specific functions on different systems, represent significant stumbling blocks to users. Toward alleviating this problem area, the use of an intermediary processor to "uniformize" interaction languages is presented. A framework for such processing is characterized in terms of the required intermediary actions and the logical capabilities needed to perform those actions. The testbed software facilities, centered on the NBS Network Access Machine, are then portrayed. Throughout, an example application, using a common command language subset to access five bibliographic retrieval systems, is described.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Bibliographic retrieval systems; command language; interaction language; language transformation; language uniformity; user-computer interaction; user-oriented system design.

18. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PAGES
UNCLASSIFIED	74
20. SECURITY CLASS (THIS PAGE)	**22. Price**
UNCLASSIFIED	$4.00

USCOMM-DC

☆U.S. GOVERNMENT PRINTING OFFICE 1980-311-046/209

NBS TECHNICAL PUBLICATIONS

PERIODICALS

URNAL OF RESEARCH—The Journal of Research of the tional Bureau of Standards reports NBS research and develop- nt in those disciplines of the physical and engineering sciences in ich the Bureau is active. These include physics, chemistry, ineering, mathematics, and computer sciences. Papers cover a ad range of subjects, with major emphasis on measurement thodology and the basic technology underlying standardization. o included from time to time are survey articles on topics sely related to the Bureau's technical and scientific programs. a special service to subscribers each issue contains complete tions to all recent Bureau publications in both NBS and non- S media. Issued six times a year. Annual subscription: domestic ; foreign $16.25. Single copy, $3 domestic; $3.75 foreign.

TE: The Journal was formerly published in two sections: Sec- n A "Physics and Chemistry" and Section B "Mathematical ences."

MENSIONS/NBS—This monthly magazine is published to in- m scientists, engineers, business and industry leaders, teachers, dents, and consumers of the latest advances in science and nology, with primary emphasis on work at NBS. The magazine hlights and reviews such issues as energy research, fire protec- , building technology, metric conversion, pollution abatement, lth and safety, and consumer product performance. In addi- , it reports the results of Bureau programs in measurement ndards and techniques, properties of matter and materials, ineering standards and services, instrumentation, and omatic data processing. Annual subscription: domestic $11; eign $13.75.

NONPERIODICALS

nographs—Major contributions to the technical literature on ious subjects related to the Bureau's scientific and technical ac- ties.

ndbooks—Recommended codes of engineering and industrial ctice (including safety codes) developed in cooperation with in- sted industries, professional organizations, and regulatory lies.

cial Publications—Include proceedings of conferences spon- ed by NBS, NBS annual reports, and other special publications ropriate to this grouping such as wall charts, pocket cards, and liographies.

plied Mathematics Series—Mathematical tables, manuals, and dies of special interest to physicists, engineers, chemists, logists, mathematicians, computer programmers, and others aged in scientific and technical work.

tional Standard Reference Data Series—Provides quantitative a on the physical and chemical properties of materials, com- d from the world's literature and critically evaluated. veloped under a worldwide program coordinated by NBS under authority of the National Standard Data Act (Public Law 96).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Services, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services, Springfield, VA 22161, in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

following current-awareness and literature-survey bibliographies issued periodically by the Bureau:

ogenic Data Center Current Awareness Service. A literature sur- issued biweekly. Annual subscription: domestic $35; foreign

uefied Natural Gas. A literature survey issued quarterly. Annual scription: $30.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $45. Please send subscription orders and remittances for the preceding bibliographic services to the National Bureau of Standards, Cryogenic Data Center (736) Boulder, CO 80303.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

SPECIAL FOURTH-CLASS RATE
BOOK