THE NAVAL MESSAGE ANALYZER

by

Richard Dee Barnhart

Project Report submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

APPROVED:

Edward A. Fox, Chairman

John W. Dickey

John Roach

December, 1992

Blacksburg, Virginia

LH
5655
V851
1992
B376
c.2

THE NAVAL MESSAGE ANALYZER

by

Richard Dee Barnhart

Committee Chairman: Edward A. Fox
Computer Science

(ABSTRACT)

The Naval Message Analyzer was developed to show that a distributed computerized message-analysis system could be built using the CODER (COmposite Document Expert/ extended/effective Retrieval) system, a testbed for artificial intelligence methods in Information Storage and Retrieval, as the foundation of the project. The Naval Message Analyzer reads messages contained in disk files and indexes them according to dates, locations and words. Words are used as indexes to documents according to headword entries in the *Collins English Dictionary*. Dates, sources and destinations are read from the header information as well as the body of each message. Header information is in one standard format, but other information is detected using clues within the text, and is further analyzed by expert modules. A database of world-knowledge is included as a repository of proper names for ships, cities, countries and abbreviations. A database maintenance facility is an integral part of the Naval Message Analyzer, so that facts can be added or changed as necessary. The system retrieves the closest matches, in descending order, to one or more ship names, locations, dates and keywords provided by the user.

ACKNOWLEDGMENTS

Many thanks to my major professor, Dr. Edward A. Fox, for the help and encouragement he has given me. Special thanks to Robert K. France for correcting various minor faults in the CODER system and for instructing me in the preparation of modules for installation in the CODER system. Thanks to the United States Navy for their support under grant number NSWC N60921-83-GA165-B047, and to Liberty University for their financial support under the Continuing Education Program for Faculty, and for the release time during the period when the work on the Naval Message Analyzer was most intense. Thanks to others who introduced me to the CODER project and helped me over a period of several years, including Mary Beth Weaver for instruction in Mu-Prolog, Qi-fan Chen for his help in data preparation, and M. Prabhakar Koushik and Jenny-Lou Menezes for their efforts on the tokenizer and user interface. Thanks to Arnold E. "Jake" Paulson and PIMS, Inc. for the use of their office space, computers, printers and copy machine, and to Dr. Frank Chimenti and the Liberty University Computer Science Department for the use of Aldus PageMaker in preparing the various drafts of this report. Last but not least, thanks to my wife Kathleen and my children Jonathan and Christine for being understanding and supportive during this endeavor.

Table of Contents

Chapter 1: Introduction 1
1.1 Problem 1
1.2 Hypothesis 3
1.3 Approach 4
Chapter 2: Literature Review 10
2.1 AI Systems (Distributed) 10
2.2 Information Retrieval 16
2.3 Message Understanding Efforts 19
Chapter 3: Previous Work: CODER and LEND 21
3.1 History of CODER 21
3.2 Design of LEND 22
3.3 System in Place at Start-up 24
Chapter 4: The Naval Message Analyzer 37
4.1 Message Structure 41
4.2 Databases, Textual Information, Clues, and Knowledge Bases 53
4.3 Expert Modules 60
4.4 Interface 66
4.5 Implementation Results 75
4.6 Prototypes of Further Work 78
Chapter 5: Production of the Videotape 80
5.1 Equipment 80
5.2 Software 80
5.3 Transfer of Screen Images to Disk 81
5.4 Chart and Diagram Production 82
5.5 Audio Production 83
5.6 Coordinating Audio with Video 84
Chapter 6: Future Work 86
6.1 Knowledge of Ships 86
6.2 Geographical Knowledge 86
6.3 Dates 87
6.4 Subject Indexing 87

6.5 JINTACCS Expert 88
6.6 Messages in Other Formats 89
6.7 Retrieval Subsystem 90
6.8 Message Understanding 90
6.9 Interface 91
6.10 People 91
6.11 Maps 91
6.12 Security 91
6.13 Weighting of Retrieval Criteria 93
Chapter 7: Conclusions 94
List of References 95
VITA 111

List of Figures

Figure 1.3.1 Some required CODER system functors 6
Figure 3.3.1.1 Overview of AIList/CED Retrieval 25
Figure 3.3.2 Data flow diagram for object-manager 28
Figure 3.3.3.1 Usage of user interface 31
Figure 3.3.3.2 Multiprompt 32
Figure 3.3.4 A message portion 34
Figure 4.0.1 Types of information 38
Figure 4.0.2 Overview of analysis subsystem 39
Figure 4.0.3 Overview of retrieval subsystem 40
Figure 4.0.4 Data-flow diagram 42
Figure 4.1.1 Sample header 44
Figure 4.1.2 Semantics of header 45
Figure 4.1.2.1 Partial grammar for JINTACCS 49
Figure 4.1.2.2 Sample message body 50
Figure 4.1.2.3 Typical data-entry format 52
Figure 4.2.2 Sample free-format message body 56
Figure 4.2.3.1 Message portion containing a date 57
Figure 4.2.3.2 Examples of dates detected 57
Figure 4.3.1 Partial grammar for date 63
Figure 4.4.1 Initial screen 67
Figure 4.4.2 User interface screen with menu 67
Figure 4.4.3 User screen with pull-down 68
Figure 4.4.1.1 User interface screen 68
Figure 4.4.2.1 List box 70
Figure 4.4.2.2 Inquiry box 70
Figure 4.4.2.3 Reply to inquiry 70
Figure 4.4.2.4 Reply to inquiry 71
Figure 4.4.2.5 Inquiry box for location 71
Figure 4.4.2.6 Reply to inquiry 71
Figure 4.4.2.7 Selection of "Maintain Databases" 72
Figure 4.4.2.8 List of maintenance options 72
Figure 4.4.2.9 Window with edit fields 72
Figure 4.4.3.1 Document inquiry window 74
Figure 4.4.3.2 List of retrieved documents 74

Chapter 1: Introduction

The Naval Message Analyzer (NMA) was programmed in an attempt to answer the problem of message analysis and categorization in the U.S. Navy. Current handling of messages involves handling by humans as well as computers. The speed and accuracy of message analysis currently depends on the humans involved. Therefore, the NMA project was sponsored as an approach to speeding up and improving the process.

1.1 Problem

The U.S. Navy sends and receives thousands of messages every day. These messages vary greatly in their subject, classification and distribution. A computer system would be helpful if it could automatically route messages to the appropriate person as well as keep a machine-readable, easily accessed database. It would be best if the system were extremely reliable and distributed in such a fashion that a local disruption in power or communications would still allow the system to function, perhaps at a lower level of efficiency.

The recent involvement of U.S. forces in the Persian Gulf and several incidents prior to the war have drawn attention to the problem of communication under battle conditions. The more gracefully a communication system can degrade, the more difficult it is for enemy actions to cut off communications between units. Part of

the communication system aboard a ship involves interconnection between computers. It is desirable to have a system where any part of the network can function at a reduced level, if necessary, but otherwise can call upon other CPUs at critical times and/or come to the aid of another CPU during idle times.

Proper routing of messages is extremely important in a military context. "...[T]he indications and warning problem is to correlate information from a variety of sources to recognize threatening or abnormal situations and to recognize enemy courses of action" and "[t]he automation of the I&W [indications and warning] process is critical to the provision of near real time (NRT) intelligence to decision makers" [LOAT88]. Routing the messages to the proper subsystem is critical for that subsystem to correlate the information. Routing is not a simple problem, and therefore requires artificial intelligence techniques.

Speed is a problem with current computerized message systems.

> "In a time critical situation in current I&W centers, analysts sometimes discard the computer aids and return to pencil, paper and direct reading of the intelligence messages. The reasons they give primarily are those of speed and accuracy. [According to] one U.S. Space Command (USSPACECOM) analyst, 'Even if the data were there, it takes too long to figure out the right keywords to retrieve them. And even if you do retrieve the data, you cannot tell how good the information is--you have to go back to the message itself to see if it came from a highly certain source, or if it is just a possible assessment.'" [LOAT88].

With multiple processors, it may be possible to solve the speed problem. With "searchonyms" [ATTA77] as outlined in section 6.9, it may also be possible to solve the problem of finding the right keyword. These problems were partially addressed in the Naval Message Analyzer project, and could be more fully solved in future efforts.

1.2 Hypothesis

The investigation was sponsored as a "proof of concept", that a distributed computerized message-analysis system could in fact be built, based on the previously-developed CODER system, and that it would be possible to extend the system in a modular fashion. This investigation also served to test the hypothesis that CODER is in fact a viable base for the development and production of systems using artificial intelligence techniques. The initial system was to include the ability to distinguish between at least two types of messages, JINTACCS (Joint Interoperability of TActical Communication and Control Systems) [LOAT88] and free-format. (Other standard message formats were not included, but could be in the future. This possibility was mentioned in a conversation with NSWC personnel.) It was also required that the system be able to index messages by dates and locations mentioned in them as well as by words in the body of the message. This required the development and application of artificial intelligence techniques to recognize geographic names as well as dates in military format.

1.3 Approach

The Naval Message Analyzer (NMA) was produced as an attempt to prove the hypotheses above. Prolog was chosen as the language for the message analysis, on a foundation of the CODER (COmposite Document Expert/Extended/Effective Retrieval) system as it was implemented at that time, using the LEND (Large External object-oriented Network Database) system for storage and retrieval of message information.

1.3.1 Prolog

The Prolog language was chosen for the NMA project because of its pattern-matching ability and for its support of rapid prototyping. Prolog is a language designed to handle definitions of objects, rules about the objects and their relationships, and answer questions about those objects [CLOC84]. The questions about objects generally take the form of filling in "blanks" in information about the objects. (This process is actually "unification".) Therefore, Prolog is used extensively in pattern matching. It is also possible to use Prolog for rapid prototyping, as in the NMA project. Rather than write large programs and databases, a small system can be written that uses Prolog to look up answers rather than go through extensive computations. Most implementations of Prolog are not noted for their efficiency. However, various modules of a large system can be tested with small Prolog modules taking the place of larger algorithmic modules. Most of the code in the Prolog modules matches input parameters with information in their databases

and does some inferences. Computation is kept to a minimum. This allows rapid writing of modules that cover a variety of test cases. When the entire system is implemented, of course, most of the Prolog modules can be replaced with faster algorithmic modules.

The Nu-Prolog [THOM87a] compiler was enhanced somewhat as part of the CODER project. As a result, the basic CODER system calls appear to be part of the Prolog language from the programmer's point of view. (See Figure 1.3.1.) Compilation requires the use of a "MAKE" file, which allows flexibility in linking object files (".o" files). Procedurally, compilation in Nu-Prolog is the same as in C. At the end of this investigation, the CODER System was ported to Quintus Prolog [QUIN91a-f].

1.3.2 CODER

The CODER system was designed as a testbed for information storage and retrieval using techniques from artificial intelligence. Users are not always sure of exact information to retrieve a document, such as dates and keywords [SALT83a]. Artificial-intelligence techniques can place more of the burden of the search on the computer, rather than on the user. The CODER system is designed to be extremely modular, which allows the programmer to write various versions of modules to implement several possible solutions to a problem. Several versions of a single module, for example the user interface, can be written to run on several different machines without changing the other parts of the system.

`myNickName(toplevel)`	Name used to refer to this server
`initSpec`	Start-up sequence (required in all servers)
`postHyp(bb,hyp)`	Posting of hypothesis hyp on blackboard area bb
`postQuestion(bb,ques)`	Posting of question ques on blackboard area bb
`answer(mm(qq,aa))`	Answer and restart point when answer "aa" to question qq unifies with variable "A" in response to question form "mm(qq,A)" that includes predicate "mm" and question "qq".

Figure 1.3.1 Some required CODER system functors.

The CODER system uses a blackboard process for communication along with several specialized modules for scheduling and planning. When the system is started, messages for the dialogue and discourse managers are posted on the blackboard. The "strategist," which is triggered by postings on the blackboard, then activates the appropriate user interface depending on the contents of the posting. The user interface normally runs on a machine such as a Macintosh which has ample utilities for menus and other user-oriented features. Some interfaces have also been written to run on ordinary "dumb" terminals, but it takes much more programming effort to achieve a given level of functionality using only ASCII text and cursor-control commands.

1.3.3 LEND

The LEND (Large External object-oriented Network Database) system is able to store many types of objects. Its methods include the ability to index on several fields and to store many types of information from small frames to entire documents. Document storage is initiated by a call that returns the document identification (ID) number for future reference. This ID can be included in other databases which serve as indexes to the documents, e.g., a frame with a date and document ID. The LEND databases were moved from one machine to another on at least two occasions during the project. In each case, the "maps" giving locations of modules of the CODER system were changed appropriately, and LEND continued to run in the same way with no additional programming effort for this project.

1.4 Overview of Project Report

After this first chapter, which introduces the problem, hypothesis and approach to a solution, the second chapter reviews the literature on similar studies. The three sections of Chapter 2 include reviews of distributed AI systems in Section 2.1, information retrieval in Section 2.2, and previous message understanding efforts in Section 2.3.

The third and fourth chapters review the foundation upon which the Naval Message Analyzer (NMA) was built and the NMA itself. The previous work included the design and construction of the CODER system in Section 3.1 and the design of the LEND database in Section 3.2. The system in place at start-up is described in Section 3.3. The fourth chapter concerns the NMA, which was built upon this foundation. The structure of the messages is explained in Section 4.1, followed by a description of the databases, textual information and knowledge bases in Section 4.2. Section 4.3 describes the expert modules; Section 4.4, the user interface; and Section 4.5, the implementation results.

The final product was a videotape to be delivered to the U. S. Navy, which is described in Chapter 5. The description of the facilities includes equipment in Section 5.1 and software in Section 5.2. The procedures used are described in the remaining four sections. Section 5.3 is concerned with the transfer of the Macintosh screen images to disk. Section 5.4 describes the production of charts and diagrams; Section 5.5, audio production; and Section 5.6, the coordination of audio with video.

Chapter 6 describes several possible directions for future work, including knowledge bases to enhance analysis, indexing and retrieval improvements, and security.

Finally, Chapter 7 states the conclusions of this investigation.

Chapter 2: Literature Review

This investigation was concerned with the use of artificial intelligence (AI) techniques, the understanding of messages and their retrieval. The review of the literature given here includes these three areas. The techniques employed in this investigation can be used to support the analysis and routing of messages as well as to recognize important items. For example, the system might analyze impending weather conditions and activate an alarm signal at affected locations to alert particular ships or bases. These techniques also provide methods of retrieving messages dealing with particular situations in historical contexts ranging in duration from hours (sudden unusual fleet movements) to months (court martials or analyses of maneuvers). A major purpose of this investigation was to show that significant data for analysis and routing could be extracted from messages and stored for rapid retrieval.

2.1 AI Systems (Distributed)

One of the principal AI techniques used in this project was that of using a "blackboard system" to distribute the processing. Literature on distributed AI systems and blackboards will be reviewed here.

Many distributed AI systems have been implemented in the past [BARR81] [BARR82] [COHE82] [BARR89]. A "distributed system" is one in which the processing takes place on several processors which may reside in the same computer or at several nodes on a network. Processing on one processor is independent from other processing. This is in contrast to

"cooperative distributed problem solving" in which the processors communicate to a much greater degree [BARR89]. These techniques are mixed in the CODER system: Certain processes in the CODER system are tightly coupled while residing on separate nodes of the network, while others are independent but may reside on the same machine. The mixture of techniques in this investigation included very few tightly-coupled processes. Therefore, the emphasis in this review is on blackboard-based systems.

A blackboard system consists of three components [NIIH89]:

- knowledge sources, that is, separate and independent components containing the knowledge needed to solve the problem;
- the blackboard data structure, a global data store which contains the problem-solving state data; and
- control over what knowledge source to apply when, and to what part of the blackboard.

The knowledge sources produce changes to the blackboard. Communication and interaction between knowledge sources occur only through the blackboard. Each change to the blackboard leads incrementally to a solution of the problem.

The reasoning behavior of blackboard systems has one outstanding characteristic: The solution to a problem is built one step at a time. The reasoning at each step may consist of any of several methods (forward or backward chaining, etc.). The control component determines which part of the emerging solution should be attended to next. Thus, "the selection and application of knowledge sources are dynamic and opportunistic rather than fixed and preprogrammed" [NIIH89].

Blackboard systems have other advantages in problem solving [NIIH89]:

- hierarchical organization of blackboard data, so that different vocabularies can be accommodated at different levels, and knowledge sources can use one level for input and another level for output;
- event-driven processing, so that any change to the blackboard is immediately made available to the controller; and
- collections of rules as knowledge sources in addition to the knowledge contained in procedures.

The presence of some or all of the following characteristics in a problem makes it a candidate for the blackboard approach [NIIH89]:

- a large solution space;
- noisy and unreliable data;
- a variety of input data and a need to integrate diverse information;
- the need for many independent or semi-independent pieces of knowledge to cooperate in forming a solution;
- the need to use multiple reasoning methods;
- the need for multiple lines of reasoning; and
- the need for an evolutionary solution.

Blackboards are currently being used mainly in rapid prototyping. The initial statement of a problem may be vague, incomplete or contradictory, and may be improved as work progresses on the system. Problems may be intrinsically ill-structured, but this is relatively rare. The blackboard approach allows the reasoning path to change as the problem is being solved. Changes to the system are localized. Design decisions may be postponed until more

is known about the problem, as various procedures and/or hardware are tested. Finally, message passing no longer requires the presence of a recipient, so that the developer of one module is freed from direct concern with the recipient of the "message" after it is placed on the blackboard. [NIIH89]

The HEARSAY-II system [ERMA80] is a well-known, early example of such a system [BARR81] [BARR89] that is used for speech recognition. The blackboard is divided into a number of levels corresponding to the hierarchical decomposition of speech into units such as phrases, words, syllables and segments. Hypotheses about units are posted on the blackboard at the appropriate level along with an indication of when the unit occurred during the utterance. The top level involves prediction, testing and concatenation of multiple-word sequences. Certain components extend words recognized at a lower level into "islands" of one or more words using knowledge about legal pairings of words. At a higher level, syntax is checked for islands of three or more words, since adjacent possible legal pairings of words may not form a syntactically correct phrase. The majority of the posted hypotheses do not appear in the final interpretation of the sentence.

VISIONS [COHE82] is a blackboard-based system for organization and control of vision systems. As in HEARSAY-II, the blackboard contains a representation of a layered description. Activation and scheduling of knowledge sources is under the control of a central executive program. The lowest layer represents regions, segments and vertices. Two intermediate levels work with surfaces and volumes, while the top two levels work with representations of objects and schemas. Hypotheses are formed concerning what the objects

must have been to result in the observed image. Schemas are frame-like constructs that may contain information about offices, airports, etc.

The CRYSALIS system [BARR82] analyzes electron density maps to deduce the structure of protein molecules. The electron density maps are three-dimensional, and analysis by an experienced protein crystallographer can take several weeks. The system must be able to infer approximate atomic locations, molecular boundaries, backbone of the polymer, etc. A blackboard architecture is again used, and the hierarchical nature of the analysis leads to a data structure similar to HEARSAY-II. Knowledge sources play a role in adding, changing or testing hypotheses on the blackboard. The control is event-driven; that is, a monitor continually refers to a list of events and triggers knowledge sources which will most probably make further headway. The monitor serves as a middle-level manager; when there is completion or failure in one direction, the monitor looks to a higher-level strategist for its next task.

HASP is a system for interpretation of sonar signals [BARR89]. HASP was designed with a blackboard architecture similar to HEARSAY-II after it was determined that the generate-and-test approach of DENDRAL was intractable [BARR89, p. 26]; the number of generated solutions was enormous. The solutions in HASP, however, can be accepted if they are less than precise, e.g., the presence of an object of unknown type in a general area. Of course, more accurate detection is desirable when possible. In contrast to HEARSAY-II, which uses a grammar to determine legal moves, HASP generates plausible hypotheses. Much of the knowledge in HEARSAY-II is in the form of procedures, while HASP uses a set of rules. Therefore, HASP is open-ended in the sense that new knowledge can be added without

perturbing the old. The major high-level task of the control system is to select the next island solution to be processed.

PROTEAN determines sets of possible three-dimensional structures of protein molecules in solution from NMR data [BARR89]. PROTEAN uses three prespecified blackboard panels, *control-data*, *control-plan* and *solutions*. Each of these panels contains multiple levels for problem subdivisions such as strategy or heuristics. The problem is described as "knowledge poor and ... underconstrained." The choice for solving such problems is generally the generate-and-test method. The strategy of PROTEAN is the application of constraints to prevent combinatorial explosion. The main function of the blackboard system is to control the problem-solving process.

TRICERO is a system for monitoring aircraft activities [BARR89]. TRICERO was based on earlier work, but is one of the first blackboard designs specifically for distributed systems. The problem is partitioned into independent subproblems, where each subproblem is solved on a separate processor.

Blackboard-based architectures have also been developed for multiple-task planning, scheduling and route planning [HAYE85a].

One of the most significant blackboard systems is the KBNL system [BARN90a] for natural-language processing. The KBNL project is an attempt to improve communication between humans and computers by having the computers use natural language. The proposed solution is to incorporate world knowledge in a world model so that when two sentences mean the same thing, the computer will derive the same meaning from both sentences. The blackboard is

used for sentence understanding as well as sentence construction for responses. This organization around the blackboard and sharing of top-level structures allows the understanding and synthesis modules to be interleaved with each other and other programs.

2.2 Information Retrieval

This investigation was largely concerned with the storage and retrieval of message information. The problem of easier and faster retrieval of information has existed for many years, and much work continues to be done at the present time.

2.2.1 Early Ideas

One of the earliest ideas for an automated information-retrieval system was the "memex" of Vannevar Bush [BUSH45]. Bush conceived of this personal information storage and retrieval device in 1932 [NYCE89], before the first feasible automatic computer was designed by John Atanasoff [DEIT85]. The importance of information retrieval was underscored later by Bush:

> "...[I]n one exceedingly important phase of the whole problem we are making little progress indeed. This is the phase of finding in the record the information that we need." [NYCE89][BUSH55].

Bush's thoughts on information storage and retrieval centered on analog rather than digital devices, whereas most work to date has involved digital techniques. Bush saw the digital approach as early as 1946, but was reluctant to get involved with the new technology [NYCE89].

Many of Bush's ideas have been incorporated into various information-retrieval systems, particularly in present version of hypertext systems such as HyperCard ®. The National Air and Space Museum is capturing documents by digitizing images and compressing them so that the required storage is "less than 100K per page," as well as using software capable of self-indexing each word or combination of words in the text [BOYN85]. Much effort has been spent on retrieval of text and the user interface as well as query formulations and measures for similarity of documents [SALT83b].

2.2.2 Database Management Systems

Another direction of specialization in information access involves databases with fields, records and indexes for data organization. This high degree of organization results in efficiency for the computer, but ordinarily requires that humans plan how to organize the data entered into the system [DATE86]. In the case of very large databases, improving efficiency is still an active area of development [MCLE90]. Even for easily-organized data, the best method of organization is still under discussion, e.g., relational [CODD90] versus entity-relationship models [RIDJ88].

There are many database programs available today. In some cases, users must learn a new querying technique or language. Even on microcomputers, development of personal or corporate databases with commercial software requires learning the peculiarities of the software [COBB87] [DUNL87] [ERIC86] [LEBL90]. Many query languages have been implemented with names such as BIPS, QUEST, CAQL and LAGRIF [SCHE82]. Some standardization of queries has been achieved, e.g., by widespread use of SQL [DATE86]

[STEE90]. The use of such languages, however, requires a large effort on the part of the novice or casual user. Attempts have been made to accept user queries in English, with some success [CODD78]. Some specialized databases require a different type of interface by their nature, e.g., large amounts of numerical or graphical information [CHEN84].

2.2.3 Document Indexing and Retrieval

Modern databases are an excellent solution to the retrieval problem when the data to be retrieved is well-defined and able to be formatted. Much information comes to us in the form of documents, however, and the indexing of this information is a large task. Bush's idea was for an individual user to make his or her own links and paths through the data, but this is not yet feasible for many individuals using common data in a large organization [NYCE89]. Much work is being done on both the indexing-linking problem [COOM90] and the user interface-retrieval problem. Ideally, an information storage and retrieval system would allow the user to retrieve data and documents according to concepts [SCHA81], which was one thrust of Bush's paper [BUSH45].

MONSTRAT was an attempt to model what a human intermediary does in an information search [BROO87]. I^3R (Intelligent Interface for Information Retrieval) [CROF87] was designed to implement this type of model, with no obvious hierarchy (as in Hearsay-II); the multiple-blackboard system uses various model and communication areas so that multiple search strategies can be accommodated as required. There seems to be no analytic way to determine what is best; complex interactions are required for query formulations and evaluations. I^3R differs from MONSTRAT in the sense that it attempts to use plausible inference [BELK87a].

Many efforts to combine the storage and retrieval of data have taken the form of expert systems. Two of the most successful have been MYCIN and PROSPECTOR [BARR82]. A tutorial system, GUIDON, is a teaching aid based on MYCIN [BARR82]. A great deal of effort was expended on building the early expert systems, but with today's expert system shells, some expert systems can be built with a few person-weeks' effort and relatively little consultation with a human expert [DOYL85]. These information systems are most useful to individuals with queries in very narrow subject areas, however, e.g. bacterial infections and searches for mineral deposits.

The CYC project [LENA90] is currently attempting to put approximately 100 million facts into a database, as a basis for deeper natural-language understanding and to overcome brittleness. This large database is being used as a platform for developing languages for expressing knowledge as well as a set of procedures for manipulating knowledge. The CYC database is being used as a base for the world knowledge in the KBNL project [BARN90a] described in the previous section.

2.3 Message Understanding Efforts

In the past, much effort has gone into the preparation of data for the computer, including the specification of the medium, format and input device [BENW76]. Forms were often used so that responses could be keyed into particular input fields. However, there is a large effort on the part of data-entry personnel to format natural-language articles or messages into a form acceptable to the database. There have been attempts at producing knowledge bases from

documents automatically [SKUC85]. Much work has been done in the area of natural-language interfaces so that users can "converse" with the computer [LEHN82]. Another area of interest is the understanding of stories [BOLC80a,b]; the FRUMP system for summarizing newspaper articles was a successful earlier effort using scripts [DEJO82]. Judging from the number of reports and papers found for the bibliography of this report, more work has been done recently on interactive natural-language interfaces than on understanding previously-entered documents (stories, articles or messages). The analysis of the input is similar, but the objectives are different: in the first instance, to produce a reasonable response from the system; in the second, to produce a summary or retrieve individual facts. Because the emphasis in this project is on message analysis and indexing, natural-language user interfaces will not be reviewed in detail.

Some work has been done in the area of case law [HAFN81]. These systems analyze and index cases for later on-line retrieval. The systems, however, do not contain all relevant sources and do not go back very far in time. They retrieve only the cases specified, and do not eliminate all non-relevant cases. In addition, commercial legal search systems are expensive (e.g., $147.00 per hour) and occasionally go down [BARN88].

One of the most successful efforts at military message understanding is the PAKTUS (PRC Adaptive Knowledge-based Text Understanding System, where PRC was taken from the corporate name Emhart PRC Government Information Systems) [LOAT88]. PAKTUS was described by B. Loatman at the second Message Understanding Conference (MUC-II) held in San Diego in 1989. PAKTUS uses a highly modular organization for ease of extension. PAKTUS contains a basic English grammar as well as a large lexical subsystem, and is able to extract information with results similar to that of the Naval Message Analyzer.

Chapter 3: Previous Work: CODER and LEND

This investigation depended heavily on the existence of the CODER system and the LEND database system described below. The CODER system provides "a testbed for exploring the value of techniques from artificial intelligence in improving the effectiveness of automatic systems for analysis and retrieval of complex documents" [FOXE91a] [FOXE87c]. The underlying programs and callable routines made it possible to concentrate on the problem being solved rather than peculiarities of the underlying computer systems. Similarly, the LEND database system made it possible to do a minimum of work to write maintenance routines for the underlying "world knowledge," and to store and retrieve documents.

3.1 History of CODER

Design of the CODER system began in late 1985, by Robert K. France and the principal investigator, Edward A. Fox [FRAN86a]. The initial implementation of the retrieval subsystem followed this design as closely as possible, especially the blackboard and strategist functions. Some of the design was modified as a result of [BELK83] and [BELK84]. The spine, consisting of the knowledge administration complex, the document database and the lexicon, was implemented next [WEAV88a].

Many graduate students have worked on the development of the CODER system. [APTE87] describes the development of the CODER communications functions. The initial user interface is described in [KHAN88], and the creation of a Prolog fact base in [WOHL86].

By 1989, a user could look up references in the AIList database of messages and find definitions of words in the *Collins English Dictionary*. Queries could be reformulated on the basis of terms found in the dictionary definitions of search terms or terms found in retrieved documents. Retrieval and user activity could proceed in parallel.

Several parts of the system reside on different machines. The C language source code has proved to be quite portable. When computers are added to the network or replaced, conversion of the code has required a minimum of effort. Because most of the machines are mounted as file systems on the others, the source modules can usually be kept on one machine and compiled on other machines from there, with appropriate changes to the "MAKE" files.

MU-Prolog [NAIS83], an interpreted language, was used in the early versions of CODER. This was replaced by NU-Prolog, which is nearly identical except that it can be compiled [THOM87b]. Later versions use Quintus Prolog [QUIN91a-f].

3.2 Design of LEND

The LEND database system is designed to be an integral part of the CODER system. The following description comes from [FOXE91a]:

> LEND is an object-oriented database system supporting abstract data types, inheritance, complex objects, semantic relations, and set-oriented and navigational access to objects. Its target applications include traditional information retrieval models, hypertexts, and SNePS style semantic networks. Based on an object-oriented model and engineered in an object-oriented fashion (entirely in C++), LEND is highly extensible for new dimensions of applications and for the enhancement of system performance. This has been made possible by our work on minimal perfect hash functions (MPHF) and order preserving minimal perfect hash functions (OPMPHF), which can find these functions for large key sets very rapidly and allow retrieval of any object stored in the network with a constant number of disk accesses. LEND has also been extended to handle hypertext/hypermedia and F^3L objects [CHEN90]. LEND has been under development since 1988 and a first version was made operational in early 1990 with approximately 70 MB of data from the CED (*Collins English Dictionary*) and the AIList news digest. The two databases showed exceptional retrieval performance as reported in [CHEN90].

LEND has a three-layer architecture. The topmost layer is the applications layer which provides users with query languages and data access. The intermediate layer provides facilities for structural abstractions. The lowest layer provides storage and retrieval of LEND objects on disk or in main memory. The support for primary memory and disk resident data in LEND is handled by two database storage managers. Both have the same message syntax to ease switching between the managers. This layer is coded in C++ as a set of classes. It uses AVL trees for primary memory resident objects and MPHFs and OPMPHFs for disk resident objects.

Only the lowest layer was used directly in this project.

3.3 System in Place at Start-up

The CODER system has been implemented on several different UNIX machines over an Ethernet. This collection of machines changes from time to time, but the fact that many modules are written in C makes them portable with very few changes. The PROLOG compilers are also written in C. The system is therefore very flexible and easily adaptable to a changing environment. Because some of the machines in the network have in fact been replaced by other machines of varying architectures during the total development of the CODER system, the system has become both flexible and portable out of necessity.

3.3.1 Overview

The CODER system was already in an advanced state of development at the start of this project. A user interface with many methods had been developed already, as outlined below, as well as a tokenizer for natural-language input. A dictionary was available, along with a morphological analyzer that could find the root dictionary entry for almost all forms of English words. The general structure of the AIList/CED search and retrieval system is shown in Figure 3.3.1.1.

A highly-developed subsystem was available for storage and retrieval of documents using the LEND database management system. The LEND user interface was not yet perfected, so the initial databases of locations and ships were entered by means of a formatted text file. The low-level LEND functions were used extensively.

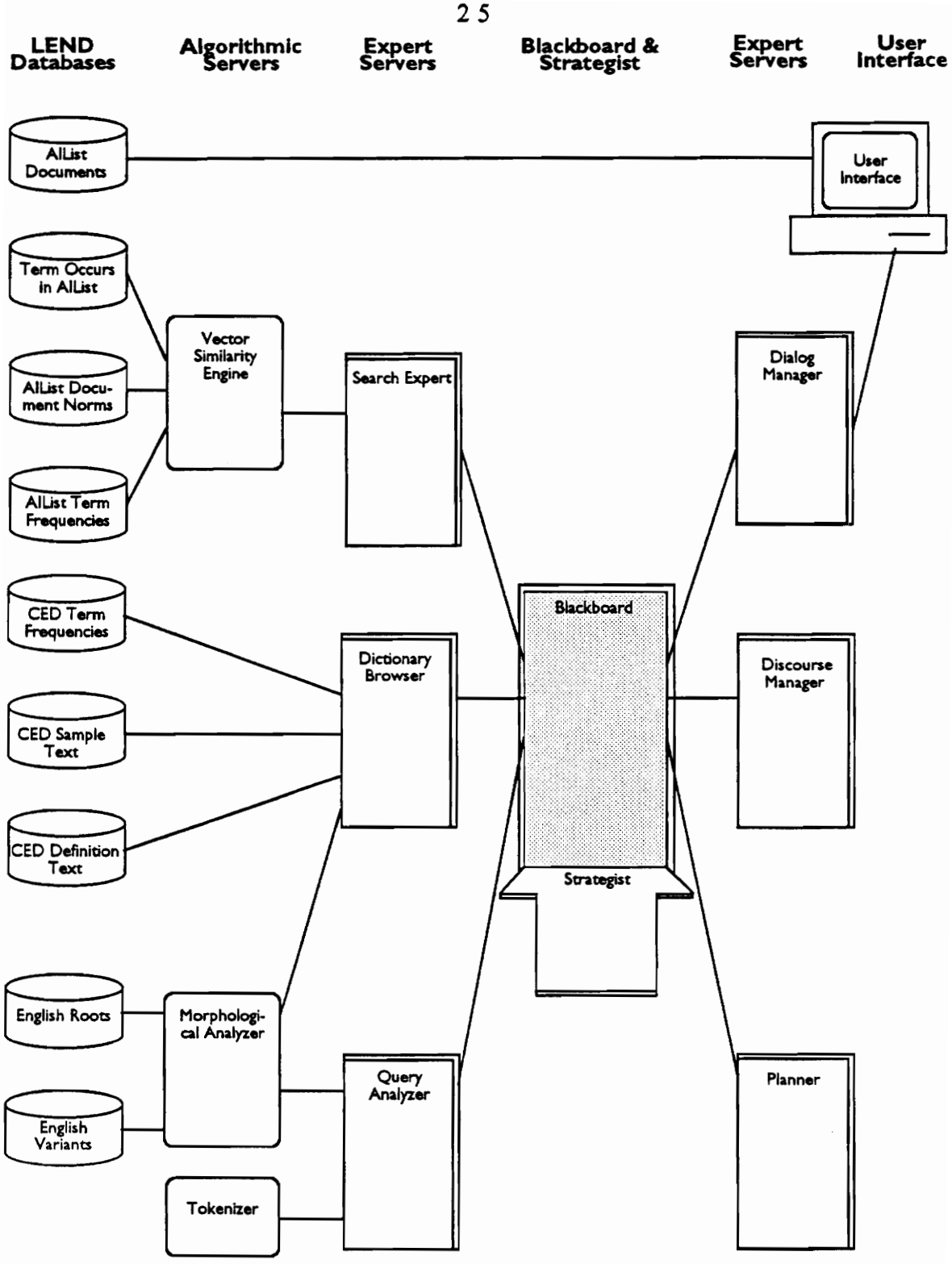


Figure 3.3.1.1 Overview of AlList/CED Retrieval System

The system was capable of looking up references in the AIList message base using several document similarity norms. It could also find the definitions of words in the *Collins English Dictionary*. The analysis of the AIList documents had taken place at an earlier time. Suffix removal had been done on the words in the document and the message base was indexed on the resulting terms. Similarly, vector queries were analyzed by suffix removal and matched with entries in the index.

There was already an extensive analysis subsystem, and the user interface had many functions in place, as described below.

There was always a perceptible time lag in the system between selection of an item by the user and the apparent action on the item. At times the selection of two items in rapid succession would cause CODER to hang or crash. The reason for this behavior was not determined precisely enough to eliminate the problem. Many concurrent activities were going on most of the time while the system was running, with no crashes.

There is a function "log" available for writing to log files. This proved most useful in tracing the execution of various modules, for this reason: The execution proceeded on several machines asynchronously, and the time stamp on each logged message allows the programmer to correlate events. The "object-manager" supervisory program directs all log messages to the directory log/progname, where "progname" is the name provided by myNickName (see Figure 1.3.1). The programs as compiled always log a start-up message and, upon normal termination, an ending message. The file name for the logged messages is the process ID assigned by the operating system. During development, it would be nearly impossible to debug Prolog programs without the ability to log messages; writing messages to the screen

only works in rare cases, since the user is not generally logged onto the same system as the process is running on.

3.3.2 System Management

The "object-manager" supervisory module keeps track of which machines have particular modules available, and the "strategist" module keeps track of which predicates should be submitted to which modules. Figure 3.3.2 shows the data flow among the managers and their rule bases. Whenever a new module is introduced to the CODER system, its location is placed in a "map" file to be read by the object-manager on the machine where the command is given to start CODER. An object-manager must be running on each machine having CODER modules. The object-managers are in communication with each other so that the strategists' requests for a particular process can be granted: if a process has not been started, the object-manager on its machine will load and start it. (Theoretically, if an object-manager stops communicating, and if the process is available on another machine, the system should be able to divert requests to the alternate machine. This in fact does happen, and, during development, can occasionally result in surprises for the programmer when an older version of a module executes. The major practical implication of this capability is that of graceful degradation, which is very important in some situations.)

Under the object-managers, the next layer of managers consists of the strategist and blackboard. The blackboard receives postings of questions and hypotheses, and transmits the postings to other modules upon request. Experts may request to "read" particular types of postings from the blackboard. The strategist uses two rule bases to determine which experts should be activated to respond to blackboard postings of questions or hypotheses. Experts

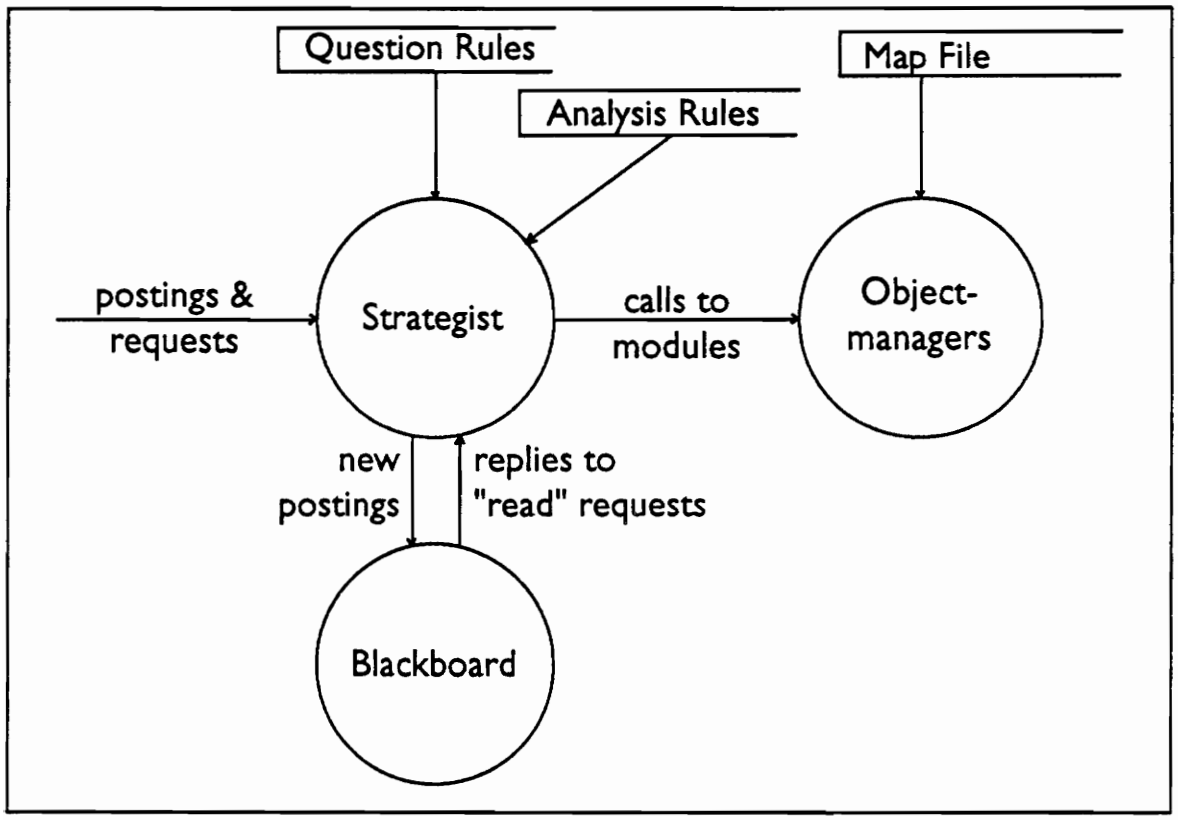


Figure 3.3.2 Data flow diagram for object-manager modules.

may request the strategist to "attend" to a blackboard area and take appropriate actions depending on the postings found there.

There is a rule base, "analysis rules," for responses to posted hypotheses. Each rule contains a predicate that serves as a "stimulus" for the strategist. Conditions may also be listed, as well as a confidence interval which must be satisfied in order for the task to be initiated. Several types of tasks may appear in the task list; usually the task is for a particular expert to prove a predicate, along with a scheduling priority. Finally, a list of actions may be included, usually "assert" or "delete."

There is also a rule base, "question rules," for posted questions. A rule contains the identification of an expert ("mynickname( )"), a relation (Prolog predicate) that the expert can process, and a processing priority. More than one expert may be capable of answering the same question, in which case all experts capable of answering the question are scheduled simultaneously, although the priorities may differ.

There is a method of direct communication between modules, by means of the "ask" predicate. The name of the called module is included in the predicate, so that the strategist need not be involved. This type of call is usually used between an expert module and an algorithmic server.

3.3.3 Interface

The Macintosh interface was coded to run under the A/UX version of the UNIX operating system. Many routines were used from the available "toolbox" set [CHER87a,b]. The

particular user interface for this project was therefore specific to the Macintosh. Another interface using the X Window System is under development, so that the interface will be more portable. It is planned that the interface will run on most machines such as DECstations that support the X Window System.

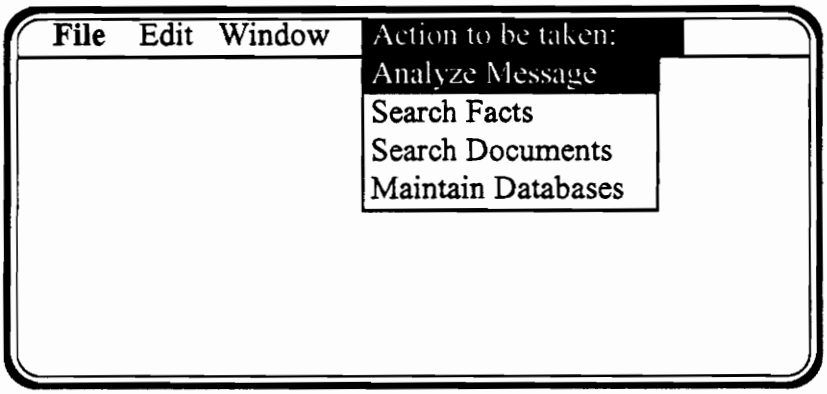
The user interface had edit fields, radio buttons and menu selections available. Each selection or item of information was presented to the user in a window on a Macintosh computer. The windows were activated by system messages from blackboard postings, and selections or cancellations of windows were posted on the blackboard for further action. The user interface also had the capability of retrieving a document when the document title or summary line was selected from the list of documents matched to the query. According to other participants in the CODER project, there was an unacceptable delay if the document retrieval was done by another module and communicated to the user interface; therefore, the actual retrieval capability was included in the user interface.

The user interface already had several methods available. A most useful method is one in which an entire list of prompts is passed to the Macintosh ("multiprompt"), and the user can then select options through edit boxes, radio buttons, etc. Each screen is formatted by the interface according to the list of parameters in the blackboard message. The reply is posted on the blackboard with the reply list as one of the parameters (see Figures 3.3.3.1 and 3.3.3.2 for input, screen and output).

Posting of Menu

```
postQuestion(newMenu(newmenu,'Action to be taken:',
                ['Analyze Message',
                 'Search Facts',
                 'Search Documents',
                 'Maintain Databases'])).
```

User Screen



Activated Predicate

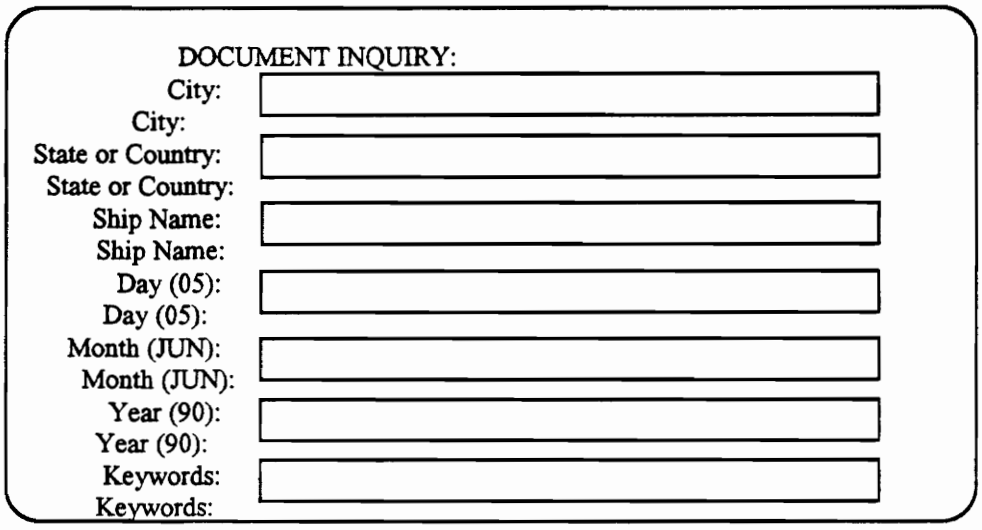
```
menuOptionSelected('Action to be taken:',
                   'Analyze Message') :-
```

Figure 3.3.3.1 Usage of user interface for menu selections.
Top: Predicate which posts question on blackboard.
Center: Format of resulting menu option when pulled down.
Bottom: Predicate activated when first menu option is chosen.

Posting of Question:

```
postQuestion(multiprompt(['DOCUMENT INQUIRY:',
                          ['31','   City:'],
                          ['31','   State or Country:'],
                          ['31','   Ship Name:'],
                          ['31','   Day (05):'],
                          ['31','   Month (JUN):'],
                          ['31','   Year (90):'],
                          ['31','   Keywords:']],
                         L)).
```

Inquiry Window:



Reply List :

```
answer(multiprompt(['DOCUMENT INQUIRY:',
                    ['31','   City:'],
                    ['31','   State or Country:'],
                    ['31','   Ship Name:'],
                    ['31','   Day (05):'],
                    ['31','   Month (JUN):'],
                    ['31','   Year (90):'],
                    ['31','   Keywords:']],
                    [["NORFOLK",
                      "VA",
                      "ENTERPRISE",
                      "",
                      "MAY",
                      "90",
                      ""]])).
```

Figure 3.3.3.2 Multiprompt.

Top: Posting of question. '31' is the code for an edit box; variable "L" is reply list.

Center: Format of window on user's screen.

Bottom: Typical reply with reply list filled in. (This is the actual reply as seen when written to a log file.)

3.3.4 Tokenizer

The tokenizer is a module that accepts a string from another process and returns a list of tokens. Each token consists of a triple: the type, the substring, and the token number. The type is a classification such as "WORD", "NUMBER", "BLANK" or "PUNCTUATION". The type allows the programmer to make a template such as token of type "NUMBER" followed by a "PUNCTUATION" type, followed by another "NUMBER" for detection of patterns for dates, names, etc. Figure 3.3.4 shows a sample message portion and its tokenized list.

The tokenizer was modified to accommodate input from a file as one of its methods. In its original form, it could accept a string and return a (Prolog or C) list of tokens. Transfer of exceptionally long strings (e.g., a long message) turned out to be inefficient, as well as at times exhausting the CODER communication space allowed for strings. Therefore, a new method was created so that a file name (complete UNIX path) could be supplied and the text read from the file. Again, the tokenizer returned a list of triples as before. The tokenizer is usually called directly rather than through blackboard postings, for two main reasons: the calling process cannot proceed without the results, and blackboard postings use many more system resources.

3.3.5 Analysis Subsystem

The analysis subsystem was capable of taking a word from another module, removing suffixes, and returning the identifier of the entry in the *Collins English Dictionary*. This

Sample Message Portion:

```
RETURN OF USS ENTERPRISE ON 12/5/89.  REPAIRS WILL BE
```

Tokenized Message Portion:

```
..., "WORD", "RETURN", 135, "BLANK", 1, 136, "WORD",
"OF", 137, "BLANK", 1, 138, "WORD", "USS", 139,
"BLANK", 1, 140, "WORD", "ENTERPRISE", 141, "BLANK",
1, 142, "WORD", "ON", 143, "BLANK", 1, 144, "NUMBER",
"12", 145, "PUNCTUATION", forward_slash, 146, "NUM-
BER", "5", 147, "PUNCTUATION", forward_slash, 148,
"PUNCTUATION", period, 149, "BLANK", 2, 150, "WORD",
"REPAIRS", 151, "PUNCTUATION", newline, 152, "WORD",
"WILL", 153, 1, "BLANK", 154, "BE", "WORD", 155, ...
```

Figure 3.3.4 A message portion and its list of tokens. The first item shown, "WORD", denotes the type of item found. The second, "RETURN", is the first word shown in the message portion. The third, 135, denotes the 135th token in the message. "BLANK" denotes the type of the next token, and so on. The number after the word "BLANK" indicates the number of spaces found.

capability was used in this project for indexing naval messages by terms in addition to indexing by dates and locations.

The morphological analyzer accepts a string, of type WORD as determined by the tokenizer, and attempts to find the root entry in the English dictionary by suffix removal. It returns the (integer) key to the root in the dictionary database. The morphological analyzer is used both in analysis (building a word index for the document) and in retrieval (matching the user's words in the query with documents).

There was a date-parsing module already in place for the CODER system. Unfortunately, this module did not recognize dates in military format, which is the format of the majority of chronological information in naval messages.

3.3.6 LEND Database Engine

A powerful database engine was also available. It is used by the morphological analyzer as well as by the various experts to retrieve lists of keywords and to match words with geographical names, titles, standard abbreviations, etc. Multiple key fields and indices can be defined.

The LEND database engine [FOXE91a] was written in C, and individual databases were located on several machines. The code is quite portable, and with relatively few modifications ran on a VAX, a Sequent and an IBM 3090 (under their various versions of UNIX). The writing of the database maintenance programs did not require any specific knowledge about the target machine on which the databases would run, once the engine was in place. In fact,

the database maintenance programs in this project were all written in Prolog, with calls to the necessary user-interface and database modules. The strategist submits blackboard postings to the appropriate machines and modules as described above.

Chapter 4: The Naval Message Analyzer

The Naval Message Analyzer (NMA) is a prototype system for analyzing military messages, some of which are highly structured and others which are not. The NMA was specifically required to distinguish between messages in JINTACCS format and messages in free format. There are other message types, such as those originating from the weather service, which are considered by the NMA to be in a free format, since the format is not defined to the system.

There are three main types of information stored in the system (see Figure 4.0.1). First, world knowledge and documents are stored in a LEND database, along with indexing information created by the system. Second, clues (patterns and trigger words) are stored in a Prolog fact base that is consulted at start-up. Finally, some specific information (on dates, for example) is stored in the logic of one or more experts.

The Naval Message Analyzer consists of a top-level program that calls the user interface and processes user replies, along with a community of expert programs. Based on the user's input, calls or blackboard postings are made to the analysis, inquiry or database maintenance subsystems. The highest-level of program in each of these subsystems in turn calls other programs (usually through blackboard postings) as required. This calling hierarchy includes several layers. Operation of the routines also depends upon the structure of the message being processed and stored. An overview of the interrelationships of the analyzer modules is given in Figure 4.0.2, and the retrieval modules in Figure 4.0.3.

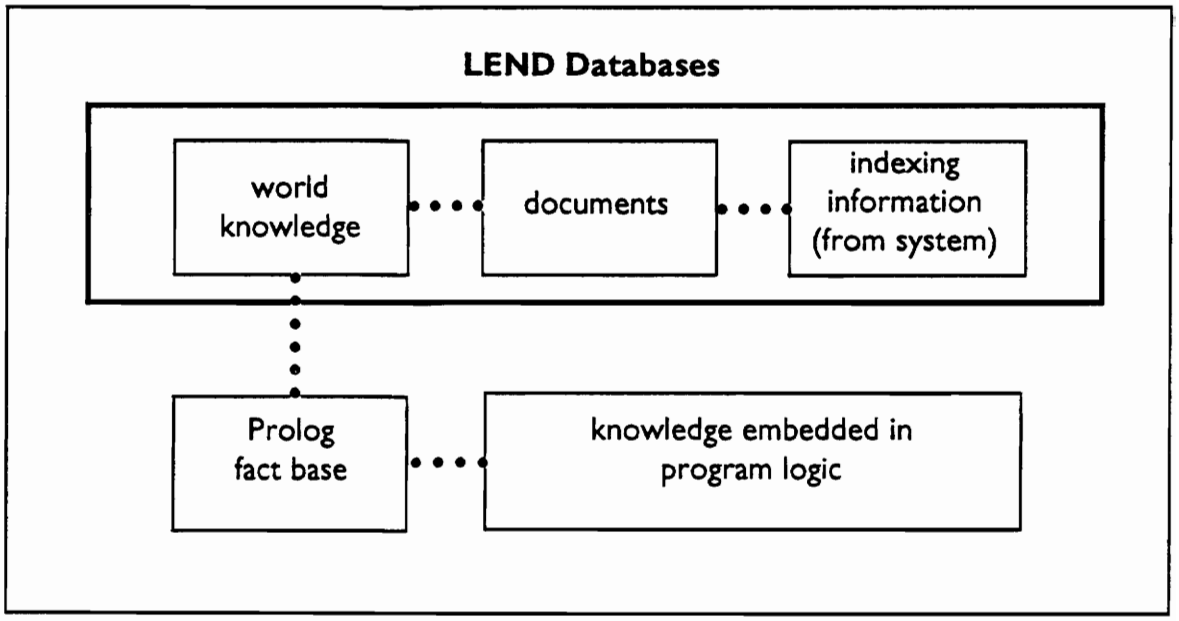


Figure 4.0.1 Types of information in NMA system.
Dotted lines indicate logical connections between information types, e.g., the Prolog fact base contains clues which trigger retrieval of facts from the world knowledge base.

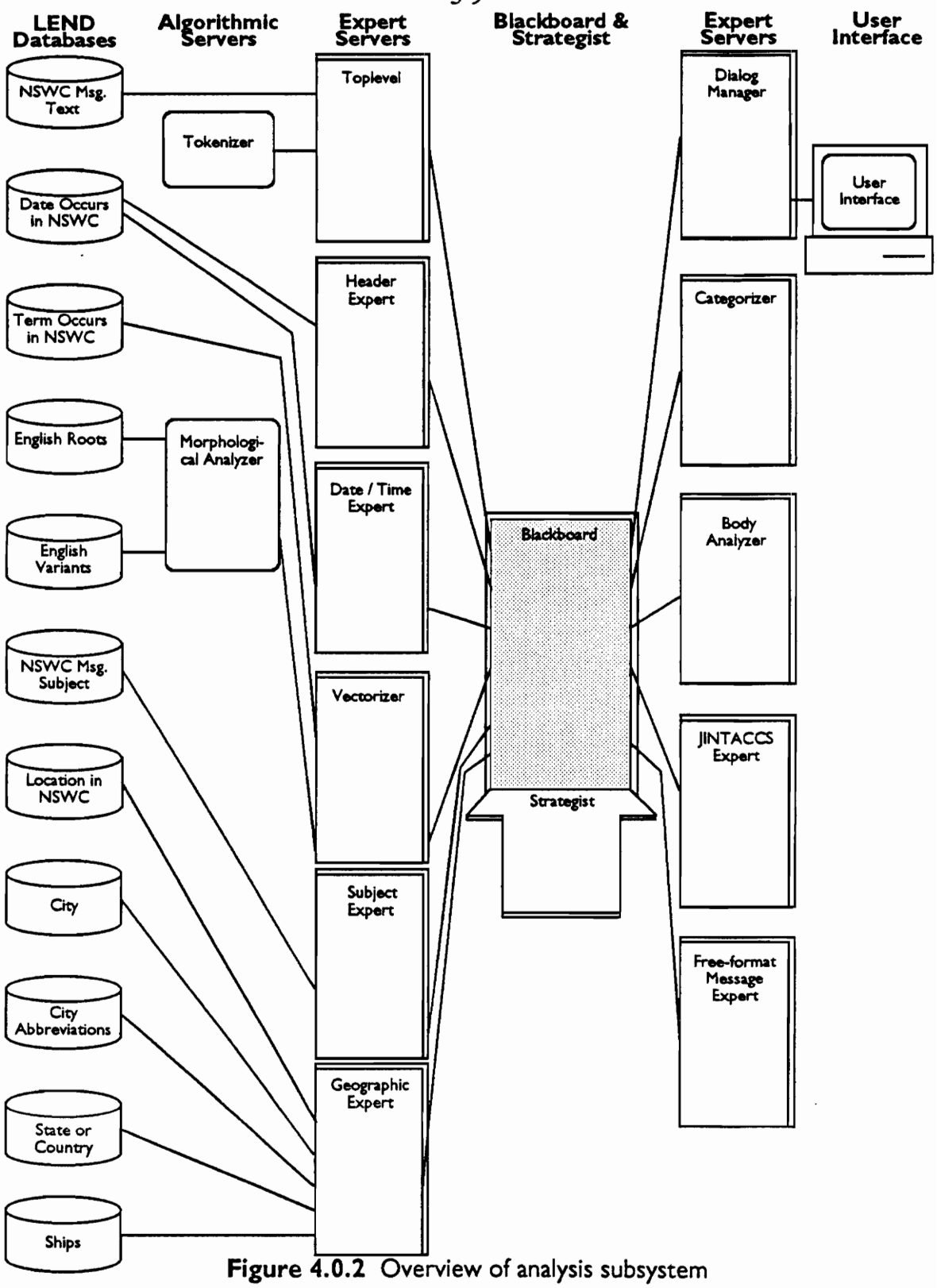


Figure 4.0.2 Overview of analysis subsystem

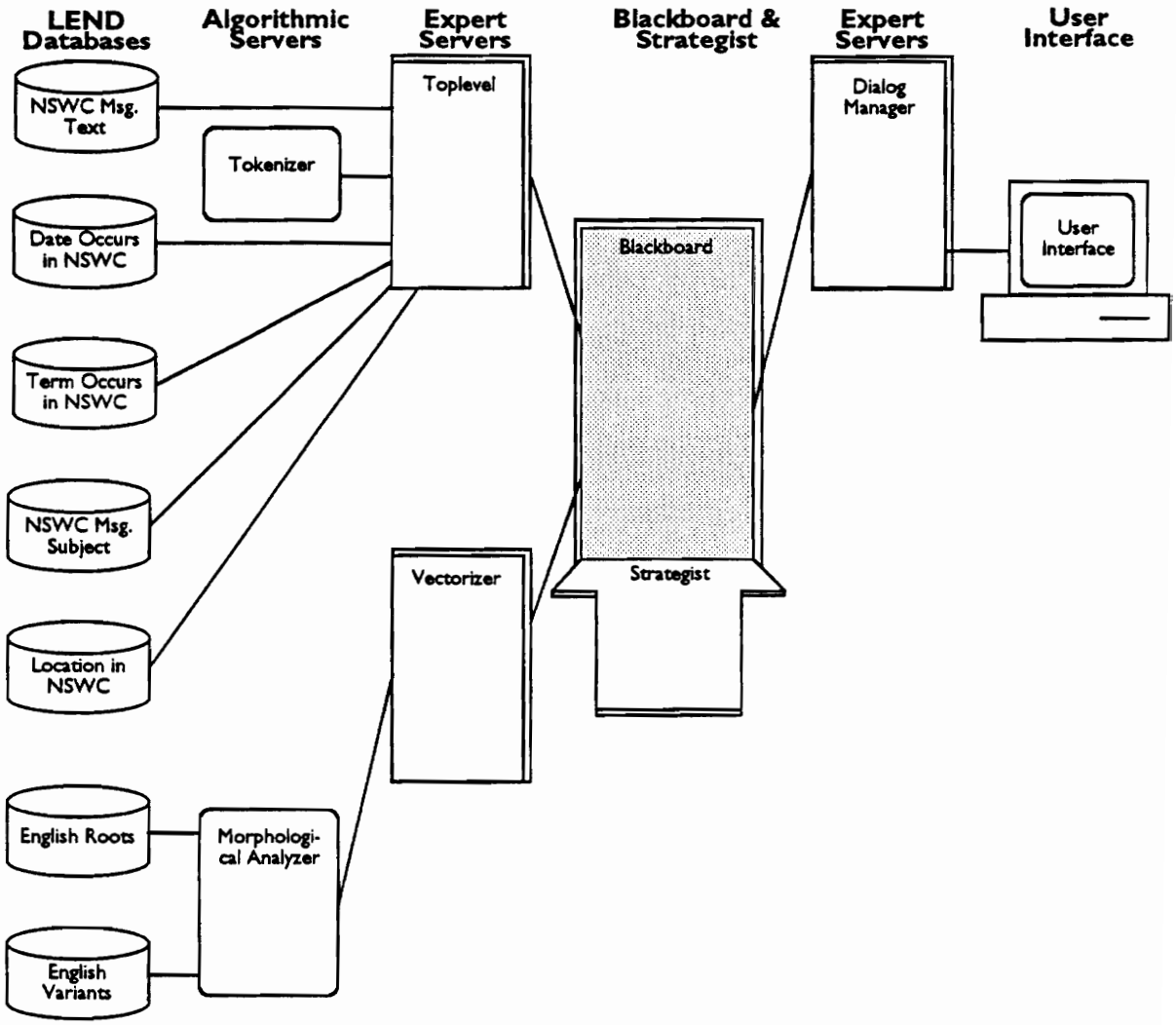


Figure 4.0.3 Overview of retrieval subsystem

A data-flow diagram is shown in Figure 4.0.4. Most data flows between modules by way of blackboard postings, with the notable exception of data to and from the tokenizer, which is called directly.

4.1 Message Structure

All details of message structure were obtained through interviews with personnel at the Naval Surface Warfare Center in Dahlgren, VA.

Each message consists of three parts: header, body and trailer. The header and body contain information, while the trailer contains only a sequence of characters denoting the end of the message, and occasionally some redundant data such as the message identification string repeated from the header. Therefore, only the header and body will be described here. Complete descriptions of message formats are available only in classified Navy documents, so the following descriptions are necessarily not definitive.

The NMA was designed to analyze the sample messages provided by the Navy. Many messages were provided, but only six in machine readable form. All others had to be keyed in by hand, so twelve more were entered into the system. The hand-entered messages were chosen to be a representative sample of the message printouts. Four more messages were made up specifically to test the expert subsystems.

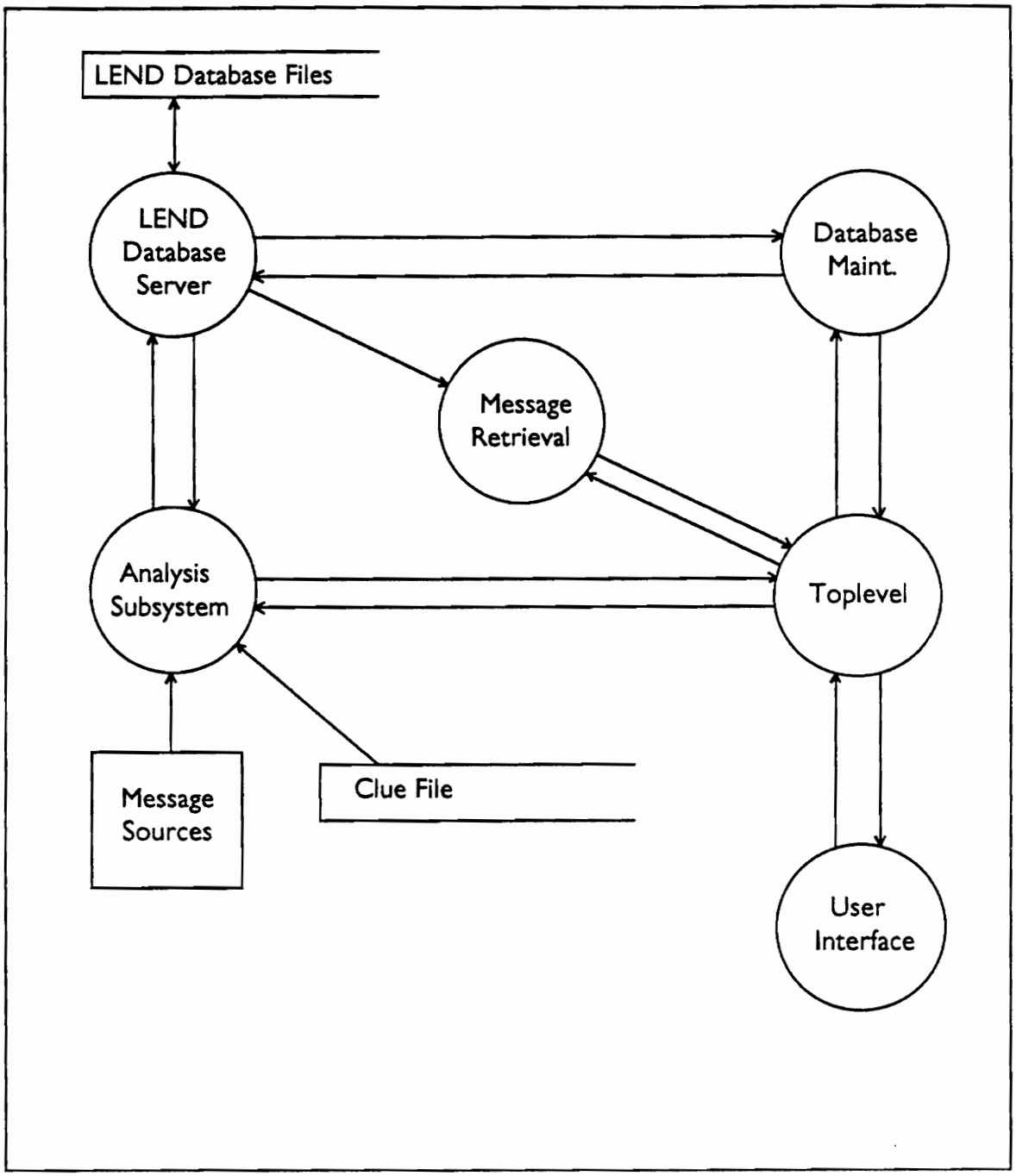


Figure 4.0.4 Data-flow diagram for Naval Message Analyzer. Most of the data is transferred by means of blackboard postings.

4.1.1 Header Format

The format of the message header is quite strict, which means that an analysis system has little difficulty with it. A sample header is shown in Figure 4.1.1, and explained in Figure 4.1.2. Note that the line numbers refer only to this particular figure; blank lines are optional, and other information may be interspersed with the information shown, as noted below. Spaces are also optional in several of the line formats, and are therefore often not significant.

Line 1 of the header is the "Start of Message" line, which contains a standard text string with the letters "ZCZC" embedded in it.

Line 2 is a string of symbols whose meaning is determined by position within the line. The first character is the priority, selected from the letters RPSZX, whose meanings range (in order) from R (Routine), the lowest priority, to X, the highest. The next two characters represent the Language Media, e.g., A = 8 level TTY (possibly from ASR = Automatic Send/ Receive), T = 5 level TTY. The fourth character is the classification, selected from UCSTZ, whose meanings range (in order) from U (Unclassified) to Z ("Eyes only"). Characters 5 through 8 form the Content Indicator or Communications Action. Following a space in position 9, the next seven characters denote the originator of the message, followed by a four-digit sequence or serial number; together with the date, these form a unique message identifier. Following the next space, there is the three-digit julian date (range: 001 through 366), and the 24-hour filing time (hours: 2 digits, 00-23; minutes: 2 digits, 00-59). After a hyphen come four characters of classification information. The classification in column 4 is repeated in column 30. The next three characters may be the same if there is no further classification information. If, for example, the message is unclassified but the distribution

```
 1   VZCZCXXX001
 2   PTTUZYUW RUWNSGG8352 2561857-UUUU--RUEBJLB.
 3   ZNR UUUUU
 4
 5   PRIORITY
 6
 7   P 121036Z SEP 89
 8
 9   FM ATKRON NINE FIVE
10
11   TO RUCOSAA/NAVAVNDEPOT NORFOLK VA       AIG FOUR TWO THREE
12   AIG SEVEN SIX TWO ZERO
13
14   INFO RHWIDHA/COMMATVAQWINGPAC WHIDBEY ISLAND WA
15   RUWFVAA/COMCARAIRWING ELEVEN            RHWIDHA/NAS WHIDBEY ISLAND WA
16
17   BT
```

Figure 4.1.1 Sample header showing some data items.

```
 Line 1: Start of Message ("SOM"), usually VZCZC or ZCZC e.g.,
VZCZCXXX001

 Line 2: abbreviated information, e.g.,
ZTTUZYRM MAGISMS0001 3441130-UUUU--ARRUEDO.
|| ||    |       |    |  |    ||    > 36-69 routing; final period
|| ||    |       |    |  |    |> 31-35 other classification info, if
|| ||    |       |    |  |    |        different from 30.
|| ||    |       |    |  |    > 30 classification UCSTZ
|| ||    |       |    |  > 25-28 filing time (24-hour)
|| ||    |       |    > 22-24 julian date (365 or 366-day calendar)
|| ||    |       > 17-20 Sequence/Serial Number (SSN)
|| ||    > 10-16 originator
|| |> 5-8 Content Indicator or Communications Action IO
|| > 4 Classification
|> 2-3 Language Media: A=8 level TTY, T=5 level TTY
|                      (historically, probably from ASR-33 TTY)
> 1 Priority RPSZX

  Line 3: In this case, repeat originator, SSN, julian date & time
DE MAGISMS 0001 3441130
> DE = "Data Entry"

  Line 4:
ZNU UUUUU
    | > Other classification info if different letter than class.
    > Classification repeated (twice)

  Line 5:
Z101130ZDEC85 ZYH
||            > ZYH if flash or immediate message
|> Date-time group (day hour min zone month year)
> Precedence (can be dual, e.g. Z X)
Usually spaces delimit these, but sometimes not.

  Line 6:
FM  Followed by sender ("from") on successive lines.

  Line 7:
TO  Followed by list of destinations, often more than one line.

  Line 8:
INFO  Followed by destination(s) for information only, not action
```

Figure 4.1.2 Semantics of header data items.
Taken from comments within program "header.nl"; "line" numbers do not refer to sample message, but rather to line types found in messages, not including blank lines.

is to be restricted to some particular group or organization, these characters may be different from the classification. (In the sample messages provided, these were frequently 'XXX'.) After two more hyphens come eight characters of routing information followed by a period.

The third line is redundant information, often repeating the originator and julian date, sometimes beginning with 'DE' for 'Data Entry'. In this particular example, it contains classification information. On occasion, both of these types of lines are present, in which case the classification appears on the fourth line of the message, sometimes with a blank line intervening. These lines have no new information content, but are present to make sure that the change of a single character (e.g., classification changed from 'T' = Top Secret to 'U' = Unclassified) will not go undetected. Often a list of abbreviations for offices appears here as a distribution list.

The fifth line in this example is again redundant, since the priority ('P' = Priority) is the first character in lines 2 and 7. Again, the redundancy is built in to avoid errors; in this case so that any human operator will not overlook the importance of the message.

The first character of line 7 is the priority, as noted above. Next comes the date and time in standard military format: two-digit day, two-digit hour (00-23), two-digit minute, time zone code ('Z' = GMT), three-letter month abbreviation, and two-digit year. There may or may not be spaces between the alphabetic and numeric characters, or between the time zone code and the month abbreviation. The code 'ZYH' may follow the year if it is priority 'X' or 'Z' (= flash or immediate message).

On line 9 is the two-letter code 'FM' followed by the originator (office, unit, ship, etc.) of the message, sometimes on successive lines as necessary.

Line 11 begins with the two-letter code 'TO' followed by the list of (one or more) destinations. The destinations may include an abbreviation for the particular office followed by a forward slash ('/'), the standard abbreviation for the installation or ship, and finally the city and state or country code as necessary. The list separator is the end of the line, or multiple spaces when the list is in two columns. The destinations are often in two columns if the list is long.

Some destinations begin with the four-letter code 'INFO', which means "For information only" and requires no immediate action. Occasionally there is a line with the two-letter code 'GR', which denotes the group count of destinations.

The two-letter code 'BT' on line 17 denotes the start of the message body. This is sometimes replaced by the three-letter code 'ZEM'.

4.1.2 Body Format

The format of the body of the message can be very restrictive if it follows the "JINTACCS" guidelines. Many messages are in free format, however, and the keywords and spacing of the JINTACCS format are missing. Still other messages may come from other sources such as the weather service, and may be in the format of that organization. If the message format is not known to the analyzer, the message is treated as free text.

4.1.2.1 JINTACCS Format

JINTACCS format has a well-defined syntax, except for fields entitled 'RMKS' (remarks), 'AMPN' (amplification), or 'NARR' (narrative). The complete details for the syntax are classified information and will not be included here; however, some of the unclassified portions will be described. A partial grammar for JINTACCS messages is shown in Figure 4.1.2.1.

Fields generally begin with a keyword at the beginning of a line. The body of a sample JINTACCS message is shown in Figure 4.1.2.2. A partial interpretation of that message follows:

Line 1: The message is unclassified.

Line 2: The exercise where it originated is designated 'BRAVE SHIELD', exercise message identifier 95.

Line 3: The message identification includes message type 'INTREP', originator '1STBN', etc.

Line 4: The message concerns heading information of some kind.

Line 5: The source of the message is a patrol, time 22:07:35 GMT.

Line 6: Subheading 1 indicates target data

Line 7: Columns DE (Data Entry reference) include a standard abbreviation for Target Type, and three fields of description.

Line 8: /01/ is the Data Entry reference. Each time '01' appears in a subsequent subheading beginning with '1', the line refers to the target described here. This target is a Medium Tank, type T-62, quantity 10.

```
<message> ::=
                <message classification> <exercise> <operation> <message data>

<message classification> ::=
                <character string> <CR>

<exercise> ::=
                EXER/ <exercise nickname> //<CR>
              | EXER/ <exercise nickname> / <exercise msg identifier> // <CR>

<operation> ::=
                OPER/ <operation codeword> // <CR>
              | OPER/ <operation codeword> / <plan originator and number> // <CR>
              | OPER/ <operation codeword> / <plan originator and number> /
       <option nickname> // <CR>
              | OPER/ <operation codeword> / <plan originator and number> /
       <option nickname> / <secondary option nickname> // <CR>

<message data> ::=
                MSGID/DISUM/ <introductory data> <disum message>
              | MSGID/SENREP/ <introductory data> <senrep message>
              | MSGID/MISREP/ <introductory data> <misrep message>
              | ...

<introductory data> ::=
                <msgid line> <ref line> | <msgid line> <ref line> <intro 1>

<msgid line> ::=
                <originator> // <CR>
              | <originator> / <message serial number> // <CR>
              | <originator> / <message serial number> / <month> // <CR>
              | <originator> / <message serial number> / <month> / <qualifier> // <CR>
              | <originator> / <message serial number> / <month> / <qualifier> /
       <serial number of qualifier> // <CR>

<ref line> ::=
                <ref mandatory data> / <ref optional data>

<ref mandatory data> ::=
                <serial letter> / <message or comm type> / <originator> / <date-time>

<ref optional data> ::=
                // <CR>
              | / <message serial number> // <CR>
              | / <message serial number> /  <special notation> // <CR>
              | / <message serial number> /  <special notation> /  <nasis code> // <CR>
```

Figure 4.1.2.1 Partial grammar for JINTACCS messages

```
 1| UNCLAS
 2| EXER/BRAVE SHIELD 95//
 3| MSGID/INTREP/1STBN 2DMAR/0611019//
 4| HEADING/INFORMATION//
 5| SOURCE/PATROL/220735Z/A1//
 6| 1TGTDATA
 7| /DE/TGTTYP/EQMT                        /EQMOD /QTY
 8| /01/MDMTK /T-62                        /-     /  10
 9| /02/LTWHL /2.5 TON TRKS                /-     /  12//
10| 1TGTMOVE
11| /DE/SPD     /DIR/ALT    /ELEV  /TGTSTATUS/TGT-ID          /CC/CAT
12| /01/    8KPH/NNW/-  /       50M/-         /-              /  /
13| /02/    8KPH/NNW/-  /       50M//
14| 1TGTLOC
15| /DE/AREA-NAME                            /LOCATION         /LQFR
16| /01/-                                    /18SVR243531      /A
17| /02/-                                    /18SUR253531      /A//
18| /DE/ACTTYP
19| /01/POSITN
20| /02/UNLOAD//
21| DECL/DOD DECLAS 951031//
22| BT
23| #0001
```

Figure 4.1.2.2 Sample message body in JINTACCS format.
A partial line-by-line interpretation is in the text.

Subsequent headings denote target movements, locations and actions. A declassification date is given at the bottom (Line 21), followed by 'BT' which denotes another 'Beginning of Text' for the trailer. The trailer in this case is the pound sign ('#') followed by the message number from the header.

Delimiters for message lines in this format do not include ASCII control characters such as carriage-return or line-feed. The delimiter between fields is the forward slash ('/'), and the end of a full line of data is the double forward slash ('//'). Each line within '1TGTDATA', for example, always has five fields, whether there is a single target type or many. The ends of the lines do not contain forward slashes; these appear at the beginning of the following line. The last line within '1TGTDATA' ends with a double forward slash, indicating that the next line will introduce a new category of data, and will not begin with a forward slash. Often the double forward slash marks the end of a line that should have more fields on it; any such missing fields are assumed to be blank. The lines are normally arranged so that corresponding fields appear in columns, with one data-entry item on each line. The field titles are often redundant, in which case they are included for human recipients only, to avoid confusion.

It is standard procedure for the operator to use a computer in message preparations. Pre-formatted screens guide the operator through data entry. A sample of such a screen is given in Figure 4.1.2.3.

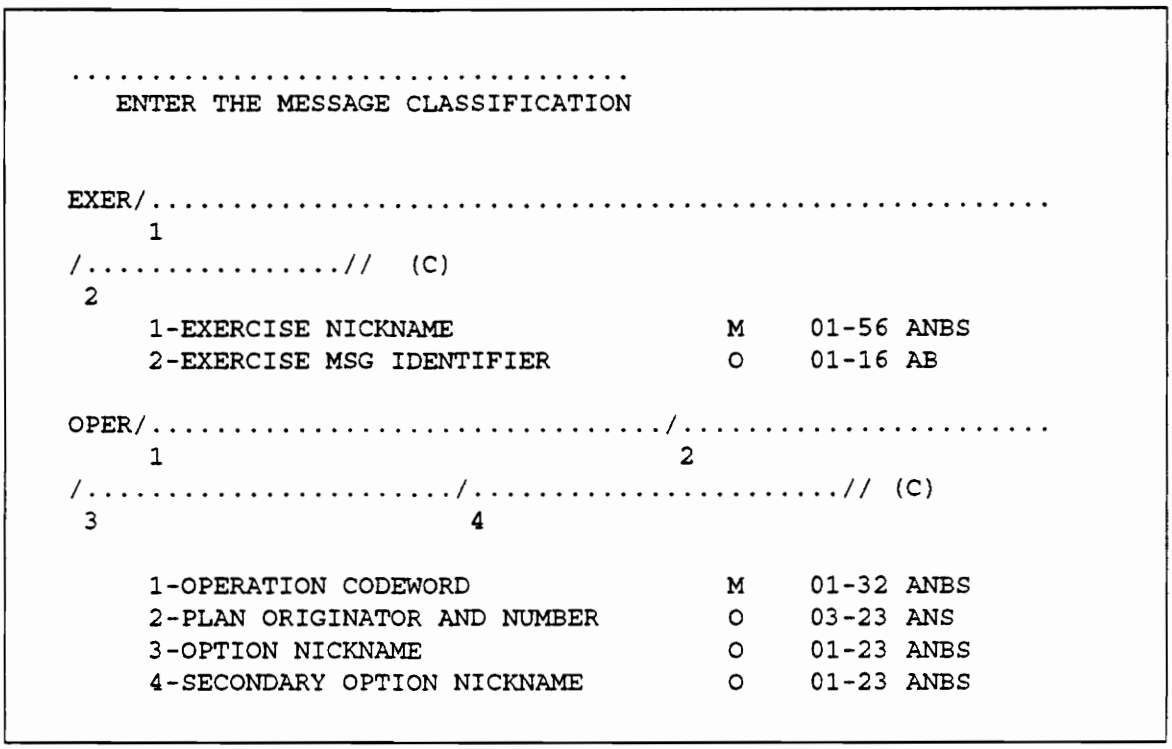


Figure 4.1.2.3 Typical data-entry format for first two lines of JINTACCS message. Each period represents one space for the operator to fill in. Numbers below each field refer to the name and format of the field appearing below the line. 'EXER/' requests entry of the exercise where the message originates. 'OPER/' requests identification of the originator. The field numbers appear directly below each field, and correspond to the list of field names below the item. 'M' denotes a mandatory field and 'O' denotes that the data for the field is optional. The ranges for the permissible number of characters appears next, followed by the character types, e.g., 'A' for Alphabetic. The entry format shown here corresponds to the first three lines of the sample message; 'UNCLAS' comes from the first line at the top.

4.1.2.2 Non-JINTACCS Messages

Much of the official communication does not fit the JINTACCS format. Among the sample messages were data on weather and sunspots, accident reports, news items on the construction of a rocket launching site in Brazil, and the announcement of the grand opening of an officers' club on one of the naval bases. Messages originating outside the Navy obviously will not be in any naval format. Other messages only fit in JINTACCS format with great difficulty, and then mostly in the spaces entitled "REMARKS".

4.2 Databases, Textual Information, Clues, and Knowledge Bases

Most of the information for world knowledge as well as all information from message analysis is stored in the LEND databases. Some of the world knowledge is stored in the form of Prolog facts also. The full text of each message is stored in a LEND database.

4.2.1 Databases

Several databases had to be defined for "world knowledge". A database of 194 ship names for the navies of the U.S. and U.S.S.R. was built using data from *Jane's Fighting Ships* [JANE89]. A geographic database was compiled from a list of 388 cities with over 50,000 population in the U.S. [GEOB88]. The standard two-letter abbreviations for all 50 states [GEOB88], and 17 two- to four-letter abbreviations for many foreign countries (gleaned from message samples) were entered into the database, as well as 15 abbreviations and variations on important city names, such as "Ft." for "Fort".

Maintenance modules were built for the various databases. These use the existing user interface, mainly through the "multiprompt" edit fields. These maintenance programs were used to add cities such as Dahlgren, VA which, in spite of their relatively small size, are important sources and destinations of messages because of nearby military installations. Other cities were added from around the world to cover most of those found in the sample message set provided. (See Figure 4.1.1). At the end of the investigation, more than 70 cities in various countries, about 50 ships, three country abbreviations, and several city name abbreviations had been added.

4.2.2 Textual Information ("free format")

There is a great variety of communication within the military, as indicated above. These messages require natural-language analysis to be fully understood. The immediate objective in this project was to extract geographical information (locations) and dates, in order to demonstrate the feasibility of adding expert system modules to the base system. Other experts can be added in the future to extract other types of data.

Although the JINTACCS expert has the potential of parsing standard fields with a great deal of precision, there is still the problem of gathering information from the free-format fields. In one conversation, naval personnel indicated that in some cases the data does not fit neatly into the fields provided, so the major part of the information lies in the explanatory remarks. Therefore, the free-format expert must be in place before the JINTACCS messages can be fully parsed and indexed. No sample JINTACCS messages were provided, however, that contained dates or geographical names in text fields.

Certain non-JINTACCS messages are also in a standard format. For example, some reports are in an outline form, where each item in the message corresponds to a blank to be filled in on a standard form. The sample message in Figure 4.2.2 is an example of a hazardous material report ("HMR"). For the purposes of this project, these messages were treated as free text, although an expert could be written to detect the form number and index the message in various ways. Automated indexing of these messages could then contribute to rapid access of problem areas such as component failures on particular aircraft or at particular bases, perhaps resulting in engineering changes and greater crew safety.

4.2.3 Clues to Information

Dates in the text are detected by consulting a table of keywords and abbreviations (e.g., "year", "JAN"), as well as by detection of patterns (e.g., NUMBER/NUMBER). A typical pattern might be

```
clue(['NUMBER',_
    ,'PUNCTUATION',forward_slash,
    ,'NUMBER',_],   datexprt).
```

For example, dates in the form 4/17 or 4/17/91 would fit this pattern, and result in a blackboard posting for "datexprt", which is the name of the date expert module described in Section 4.3. The text portions posted include 10 tokens (triples) before and after the location of the clue.

```
UNCLAS  //N04790//

SUBJ:  HAZARDOUS MATERIAL REPORT (HMR)/ENGINEERING INVESTIGATION (EI)

A.  OPNAVINST 4790.2E
B.  NA-01-85ADF-4-12 DATED 01 SEPT 1984 / RAC 2 DATED 01 NOV 1988

1.  ATKRON NINE FIVE/09707

2.  NAVAVNDEPOT NORFOLK VA

3.  R09707-89-0043

4.  9254 / NAS WHIDBEY ISLAND WA

5.  5925-01-116-7032

6.  CIRCUIT BREAKER

7.  UNK

8.  MS 14154-7-1/2

9.  THROUGH 13. UNK

14. THROUGH 15.  N/A

16.  A.  A-6E
     B.  N/A

17.  $98.06 / 2 MAN HOURS

18.  N/A

19.  WD

20.  74180

21.  HOLDING IN VA-95 QUALITY ASSURANCE

22.  A.  WHILE TROUBLE SHOOTING DRS COOL DOWN DISCREPANCY AFTER
     FLIGHT, CIRCUIT BREAKER 6073 (REF B, PG. 2-38A, FIG. 12-13,
     IT. 2) WAS FOUND DISINTEGRATED AND WIRES WERE CHARRED.  THE
     CHARRED WIRES HAD ALSO SHORTED OUT CIRCUIT BREAKER 6074 (REF B
     PG. 2-38A, FIG. 12-13, IT. 3).  CAUSE UNDETERMINED.  REQUEST
     ENGINEERING INVESTIGATION.
B.  POSSIBLE LOSS OF AIRCREW / AIRCRAFT
C.  0 IN THE LAST 12 MONTHS
D.  GROUND CREW TROUBLESHOOTING
E.  N/A
F.  NONE
G.  THROUGH H. N/A
I.  J. VESTERMAN / LT. / QAD / 820-2662
J.  A-6E/159180
K.  N/A
BT
```

Figure 4.2.2 Sample free-format message body.
In this particular case, the numbered items and lettered sub-items correspond to blanks in a standard form for reports. For reasons of efficiency, the entire form need not be transmitted to the receiver.

Sample Message Portion

```
RETURN OF USS ENTERPRISE ON 12/5/89.  REPAIRS WILL BE
```

Tokenized Message Portion

```
..., "WORD", "RETURN", 135, "BLANK", 1, 136, "WORD", "OF", 137, "BLANK",
1, 138, "WORD", "USS", 139, "BLANK", 1, 140, "WORD", "ENTERPRISE", 141,
"BLANK", 1, 142, "WORD", "ON", 143, "BLANK", 1, 144, "NUMBER", "12", 145,
"PUNCTUATION", forward_slash, 146, "NUMBER", "5", 147, "PUNCTUATION",
forward_slash, 148,  "PUNCTUATION", period, 149, "BLANK", 2, 150,
"WORD", "REPAIRS", 151, "PUNCTUATION", newline, 152, "WORD", "WILL",
153, 1, "BLANK", 154, "BE", "WORD", 155, ...
```

Pattern for Date Detection

```
clue(['NUMBER',_, 'PUNCTUATION',forward_slash, 'NUMBER',_],
            datexprt).
```

Figure 4.2.3.1 Message portion containing a date, along with its list of token triples and the pattern template (clue). The clue fits the portion of the message starting at token number 145.

Example Date	Detected	Parsed and Formatted for Index
12/5/1989	Y	Y
12/5/89	Y	Y
12/5	Y	Y
5 DEC 89	Y	Y
5 DEC 1989	Y	Y
5DEC89	Y	Y
DEC 5, 1989	Y	Y
050430Z DEC 89	Y	Y (Includes time)
050430ZDEC89	Y	Y (Includes time)
December 5, 1989	Y	Y
the fifth of December	Y	N
next Tuesday	Y	N
January of next year	Y	N
three days from then	Y	N

Figure 4.2.3.2 Examples of dates detected. Detection only requires a keyword such as "December" or "day" to be present, or a pattern such as number/number. Parsing by the date expert is not always successful because of the many ways of expressing time and date information.

This number is arbitrary; without a detailed study, it resulted in detection of all sample dates in the targeted patterns. Figures 4.2.3.1 and 4.2.3.2 show a tokenized message portion in which a pattern fits the clue above, along with examples of dates detected.

In a similar fashion, place names are detected by consulting a database of place names. In the context of this project, capitalization is not an appropriate criterion for proper nouns, since most of the messages in the sample were capitalized throughout. The technique of searching a database for each word is inefficient due to the large number of potential names to be checked for each word. However, this technique was used for two main reasons. First there are many ways place names can be expressed, including abbreviations such as "WASH" for Washington, DC and the inclusion or exclusion of state or country as in "WASHINGTON DC" and "WASHINGTON." There is also the fact that nearly any word and many phrases may be a place name when used as a noun. For example, the word "NEEDLES" (expressed in all capital letters) may refer to items required in sewing or to a city in California. An entire phrase such as "LITTLE ROCK" or "NEW HAVEN" is a geographic name, but each word taken separately has its own dictionary definition. If a phrase is detected which is probably a location (such as "Lake" or the actual name of a place), a phrase bracketing the detected word is passed to the geographic expert described below. In the case of names preceded by a ship prefix ("USS", "HMS"), the phrase is passed to the ship expert.

Finally, dates containing only month and day information should be parsed and used for indexing, as mentioned above. The year should be assumed to be the same as the message date. There were no messages in the test sample which contained only the month and day.

4.2.4 Knowledge Bases

The databases for world knowledge contain a few facts on over 100 ships and over 500 locations. Several ships and locations were added as the project progressed, both in order to test the database maintenance modules and so that the geographic and ship experts could analyze more messages. The basic information on cities and military bases is simply the name of the city or base along with the standard (two- to four-letter) military abbreviation for the state or country. A separate database of abbreviations was also built, so that "FT. MEADE", "FORT MEADE", "FT. GEORGE MEADE", "FT. GEO. G. MEADE" and any other abbreviations could be recognized, along with the full name, "FORT GEORGE G. MEADE." (All of these versions of this name were contained in the relatively small sample of messages provided by the Navy for analysis.)

The basic information on ships was name, number, type, class, and short description, such as TEXAS; 39; CGN; VIRGINIA; GUIDED MISSILE CRUISER. All information on the ships in the sample was taken from *Jane's Fighting Ships* [JANE89].

Some of the world knowledge was embedded in the programs themselves, such as the parsing for various forms of dates. A file of "clues" is provided to the main parsing program. These clues consist of both words and patterns stored as Prolog predicates as shown in Figure 4.2.3. The presence of one or more of these words or patterns results in a blackboard message being posted for the date expert.

During the analysis, indexing of the documents was done by entering records into appropriate databases. No maintenance modules were constructed for these databases, since it was assumed that much of the information would not be intelligible to casual users. For example, for each word in the document, a numeric reference to a dictionary entry was generated by the morphological analyzer and paired with the numeric document reference assigned by the system. This pair of integers only has meaning to the user when the corresponding dictionary entry and document are retrieved.

4.3 Expert Modules

After being tokenized, each message is examined by a module that subdivides it into header, body and trailer. The key to this subdivision is the sequence "<CR>BT<CR>" (carriage return, letters "BT" for "Begin Text", and another carriage return). This sequence occurs at the beginning of the body and at the beginning of the trailer. Each message portion is posted on the blackboard for processing by the appropriate expert. Detection of certain patterns in the body of the message determine that the JINTACCS expert will be called rather than the body expert.

4.3.1 Header Expert

The header expert receives the blackboard posting as a list of tokens. As it "walks down" the list, it determines whether each line of text falls into certain patterns. The strict formatting of the header lines in all messages means that there is not a great variety of possible patterns. For example, line 2 of the header must start with one of the characters RPSZX (see above) and the fourth character must be one of the characters UCSTZ. The line must also contain

a valid date as well as punctuation in certain places. This allows the header expert to determine the priority (RPSZX), the classification (UCSTZ), the message date, message ID and other facts from this one line. Sometimes the date contains spaces, but at other times it does not.

The most difficult part of the header is the destination section. In many cases, the list of recipients is in two columns so that the <CR> is not the only delimiter between locations. Therefore, each line must be examined for more than one location listed. To further complicate matters, there is frequently a naval abbreviation for a particular office or officer (e.g., "RULYOGB/CTU") preceding the geographic name. For this particular project, these abbreviations were ignored, since no list was supplied by the Navy. The header expert simply skips over unknown "words." Most of the time these abbreviations were followed by a forward slash, but not always. In addition, some destinations consisted entirely of abbreviations such as "AIG 10654;" it was not clear that there was any consistent way to distinguish between an office or officer (e.g., "RULYOGB/CTU") and a destination (e.g., "AIG 10654"). More information would have to be provided to the programmer in order to include such destinations in the index.

The header expert was written with the capability of determining many facts concerning the message, such as priority, classification, dates and locations, but only contributed dates and locations to the indexes. For test purposes, the facts were logged, using the CODER logging facility described in section 3.3.1, as they were determined. In actual use, each document's classification would be included as a separate field in the document database, and only users at secure locations with appropriate clearances would be allowed to access the document. This would require a modification of the LEND document storage and retrieval routines, so that adequate security could be introduced into the system.

4.3.2 Subject Expert

The subject expert is the smallest among the experts described here. It simply takes any message line beginning with "SUBJECT:" or "SUBJ:" and places the remainder of the line in the subject database. Subjects often contain keywords that can be indexed further, but these were not processed further by this expert. No messages were furnished in which the subject required more than one line, so this expert does not have a multiple-line capability.

4.3.3 Date Expert

The date expert analyzes the series of tokens from the blackboard posting, centered on the clue described in the section on "Clues to Information" above. Dates are all converted to one standard format for inclusion as keys to the document database. A context-free grammar for the parsing of dates is shown in Figure 4.3.3; however, the context-free information is not sufficient to determine the validity of a date. Therefore, a partial context-sensitive grammar is also shown in Figure 4.3.3.

There are several formats which the date expert can currently decode. Occasionally, a form such as M/D appears (e.g., 12/5) or DM (e.g., 05DEC), and these are evaluated after the other possibilities have been exhausted. Longer phrases are dealt with first, followed by shorter phrases; if there is a longer phrase that can be decoded, it will contain more information than the shorter phrase.

Context-free grammar for date:

```
<date> ::= <military-date> | <civilian-date>

<military-date> ::= <day-time-group> <zmonth> <year> |
                    <day_time_group> <zmonth> |
                    <day_number> <military-month-name> <year> |
                    <day_number> <military-month-name>

<day-time-group> ::= <day-number> <hour> <minute> |
                     <day-number> / <hour> <minute>

<zmonth> ::=         <zone> <space> <military-month-name> |
                     <zone> <military-month-name>

<day-number> ::=  1 | 2 | ... | 9 | 01 | ... | 09 | 10 | ... | 31

<hour> ::=        00 | 01 | ... | 23

<minute> ::=      00 | 01 | ... | 59

<zone> ::=        Z | Z1 | ... | z23

<military-month-name> ::= JAN | FEB | ... | DEC

<day_of_week> ::= SUN | SU | SUNDAY | MON | M | MONDAY |
                  TUE | TU | TUESDAY | WED | W | WEDNESDAY |
                  THU | TH | THURS | THURSDAY | FRI | F | FRIDAY |
                  SAT | S | SA | SATURDAY

<month-name> ::= <military-month-name> | SEPT |
                 JANUARY | ... | DECEMBER

<civilian-date> ::= <month-name> <day-number> <year> |
                    <month-name> <day-number> |
                    <day-number> <month-name> |
                    <month-number> / <day-number> / <year> |
                    <month-number> - <day-number> - <year> |
                    <month-number> / <day-number> |
                    <month-number> - <day-number>
```

Sample context-sensitive information:

```
<day-number> <month> ::= 01 <month> | 02 <month> | ... | 28 <month>
<day-number> FEB <leap-year> ::= 29 FEB <leap-year>
<day-number> <non-FEB-month> ::= 29<non-FEB-month> |30 <non-FEB-month>
<day-number> <31-day-month> ::= 31 <31-day-month>
<non-FEB-month> ::= <31-day-month>|APR |APRIL | ... |NOV |NOVEMBER
```

Figure 4.3.1 Partial grammar for date analysis, including context-sensitive information.

First, the Navy format for dates is D M Y where D is the day number, M is the standard three-letter abbreviation for the month name, and Y is the year number, e.g., 01 JAN 91. This pattern, ("NUMBER", "BLANK", "WORD", "BLANK", "NUMBER") is straight-forward to decode, given a table of month names and corresponding month numbers, e.g., month('JAN','01'). Even this standard format has variants since the year may be written as 91 or 1991, and the date does not always contain spaces, e.g., 01JAN91. Therefore, the presence of spaces is not significant, and they are ignored. If the value of the year is less than 100, 1900 is added.

The second form of the date is M/D/Y, where the month M is numeric rather than alphabetic. Because all information is in strings, the table contains alternate forms of the numbers, such as `month('JAN','1')` that form an adequate set to unify with the examples found in the sample messages.

The form M D, Y (e.g., DEC 5, 1989) is checked next. The full names of the months are also stored in the table (e.g., `month('JANUARY','01')`) so that month names may be found when spelled out (e.g., DECEMBER 5, 1989).

At times, the full form of the date and time has the day in the same token as the time of day. For example,

050430Z DEC 89

contains the time of day in the first four digits 0504 (5:04 AM), and the day of the month appears as the last two digits, in this case 30. The letter "Z" represents the time zone, in this case Greenwich Mean Time or "Zulu" time in naval parlance. Sometimes the time zone is not separated from the name of the month, as in

050430ZDEC89

where the four characters "ZDEC" appear in a single token and must be separated before attempting a table lookup. This is the only instance in the sample messages where this separation was necessary, since other time zones are designated with "Z" and a numeral, such as "Z5" for a time zone five hours later than GMT. Zones other than "Z" did not occur in any of the sample messages.

4.3.4 Geographic Expert

The geographic expert accepts a series of tokens from the blackboard posting and looks up phrases in the geographic database. The search starts with phrases of length 5, such as "THE PEOPLES REPUBLIC OF CHINA", and proceeds to phrases of length 4, etc. The reason for doing the lookups in decreasing order of length is that the name of one location could be contained in another name, such as "CHINA" contained in "CHINA LAKE, CA", similar to the algorithm for computing dates above.

Whenever a geographic clue is found, ten tokens on either side of the clue are passed to the geographic expert. In "THE PEOPLES REPUBLIC OF CHINA" for example, there are nine tokens in the entire phrase, including the spaces. Depending upon where the clue is detected in the phrase (e.g., first word or last word), there could be up to eight tokens in the name preceding or following the token number. Ten tokens before and after allows for up to two other items such as punctuation, as in "THE PEOPLE'S REPUBLIC OF CHINA."

4.3.5 JINTACCS Expert

The JINTACCS expert consults a Prolog fact base containing each heading type given in the JINTACCS specifications. Because the guidelines for this project were to recognize only dates and geographic names, little was done with the rest of the information. The program was, however, developed to the point that it logged each occurrence of such a category (see logging facility in section 3.3.1). Dates and locations occur only within particular items. Dates and locations, however, are sent to the "date" and "geographic" experts whenever found. No JINTACCS messages were provided that had geographic locations in the body, since they all involved exercises of a local nature.

4.4 Interface

A user interface already existed for the AIList/CED project (see Section 3.3, System in Place at Start-up). Capabilities existed for adding menus, adding or deleting menu items, and activating or deactivating menu items. On the Macintosh screen, active items are printed in black, inactive items in grey. As mentioned previously, windows can be created with edit boxes, radio buttons, and menu items.

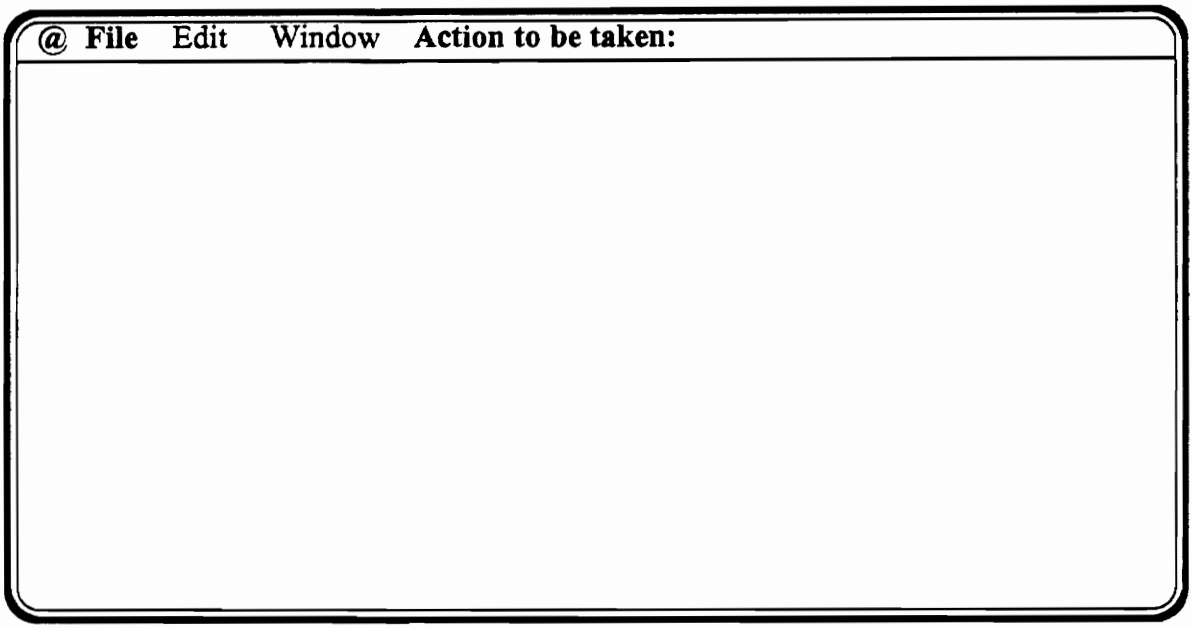


Figure 4.4.1 Initial screen for user interface. @ represents the "Apple" trademark.

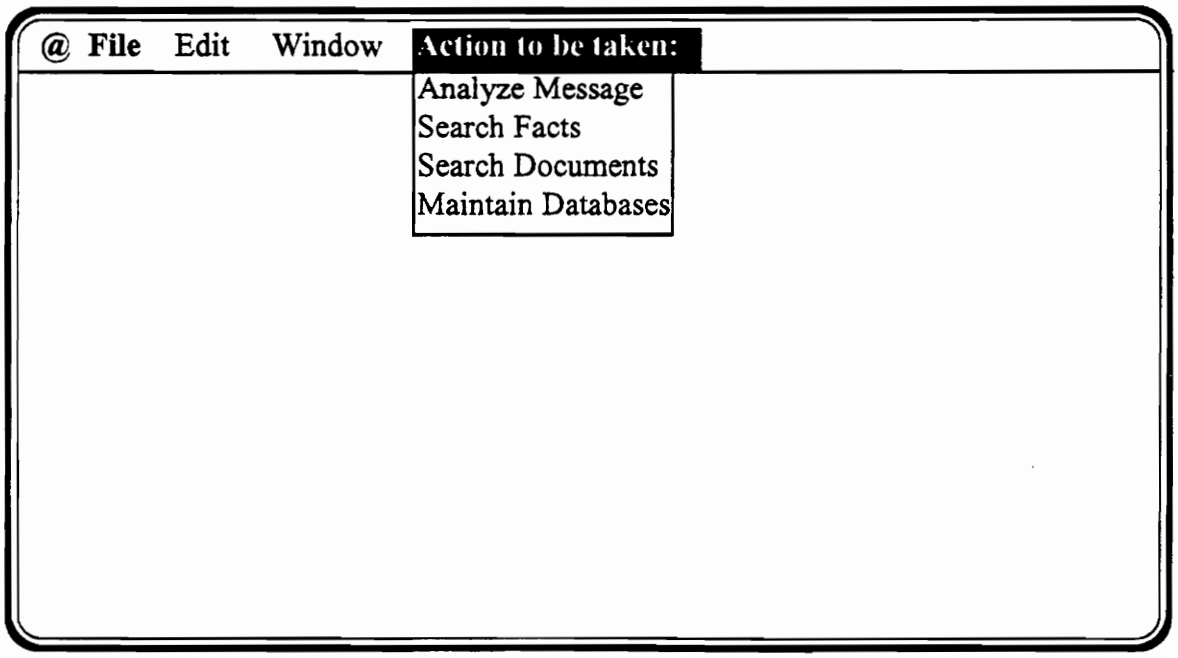


Figure 4.4.2 User interface screen with menu pulled down.

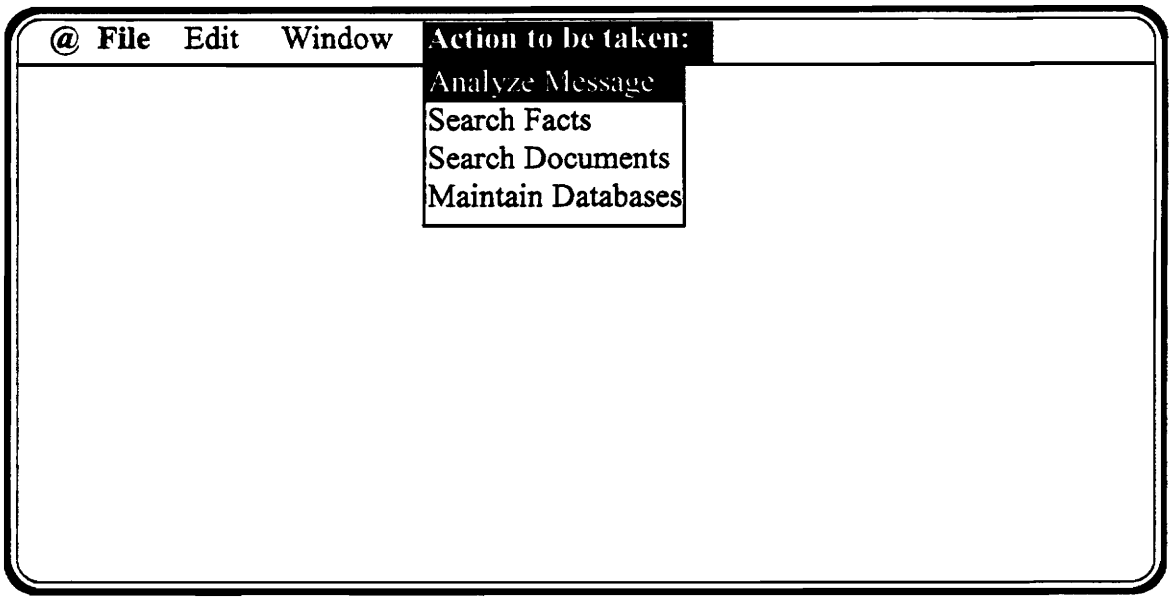


Figure 4.4.3 User screen with pull-down menu option selected.

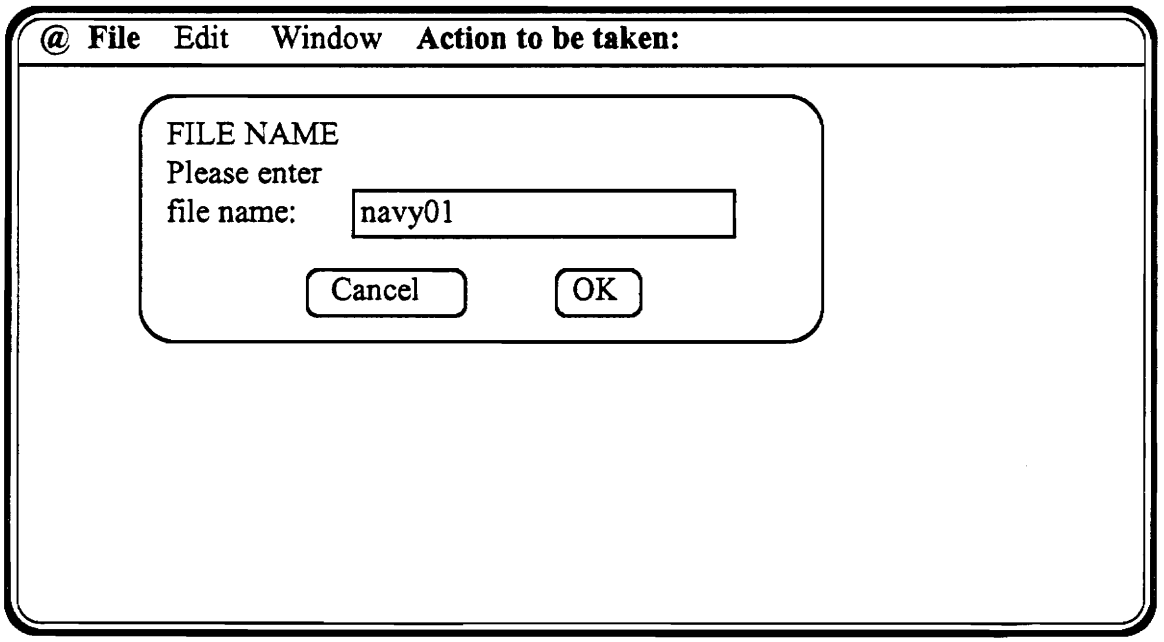


Figure 4.4.1.1 User interface screen showing edit box for file name.

The user interface is initiated with only a menu bar across the top. The only active options are "File" and "Action to be taken", as in Figure 4.4.1. The active options allow menus to be pulled down as illustrated in Figure 4.4.2. Figure 4.4.3 shows the selection of the request to analyze a message, described in the next section.

4.4.1 Analysis Request

The request to do analysis of a particular document is made by selecting the main menu option "Analyze Message" under "Action to be taken:" and entering a file name on the screen. The edit window is shown on the screen in Figure 4.4.1.1. The file name is automatically concatenated to the rest of the path to the file.

4.4.2 Database Inquiry and Maintenance

For database inquiry, the user can select a menu option to "Search Facts." The window in Figure 4.4.2.1 appears listing the current general categories available: at present, ships and locations. This general categorization is adequate to allow flexible inquiries as explained below.

The selection of "Ships" causes the window in Figure 4.4.2.2 to appear. If "NEW JERSEY" is entered, for example, all database records with that name are retrieved, as in Figure 4.4.2.3. If "38" is entered, all ships with that number are retrieved, as in Figure 4.4.2.4. The LEND database returns a set of records that match a given query, so that all matching items can be displayed at once. The selection of "Locations" results in similar windows; the appearance is similar, as shown in Figs. 4.4.2.5 and 4.4.2.6.

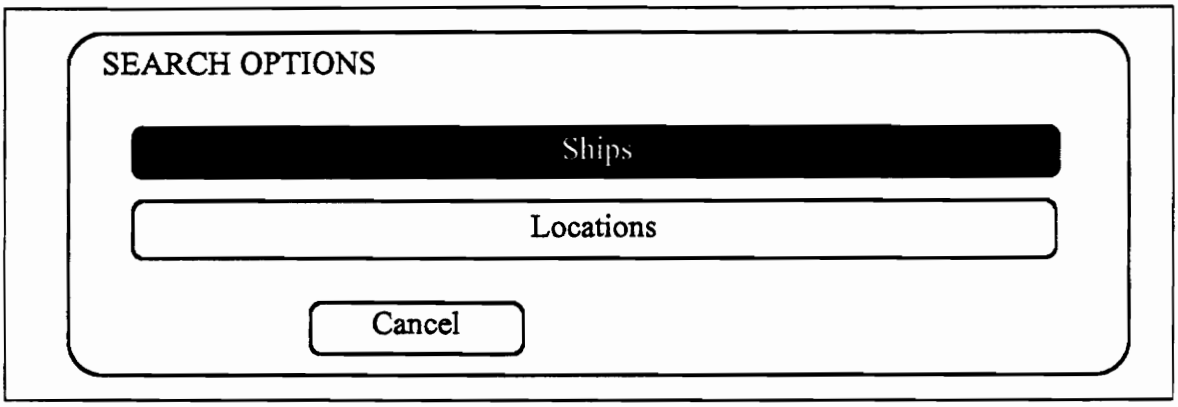


Figure 4.4.2.1 List box with two options on list, for inquiry into the two main databases.

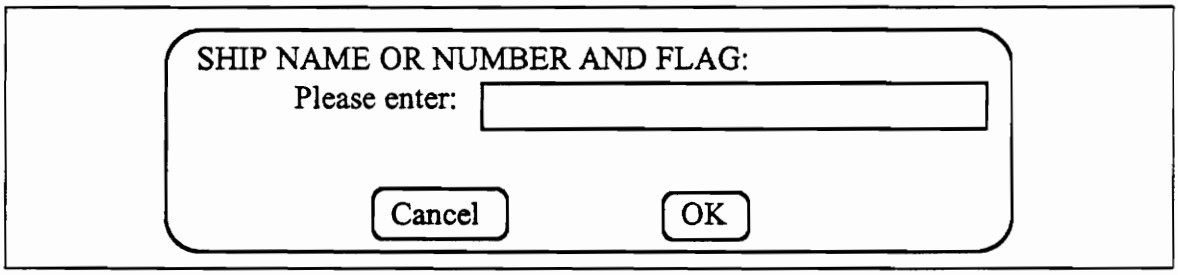


Figure 4.4.2.2 Inquiry box for ship database.

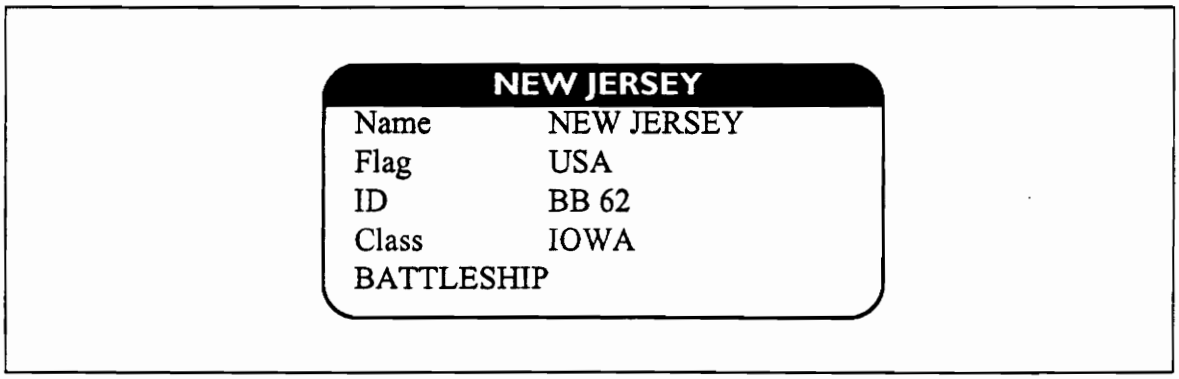


Figure 4.4.2.3 Reply to inquiry "NEW JERSEY".

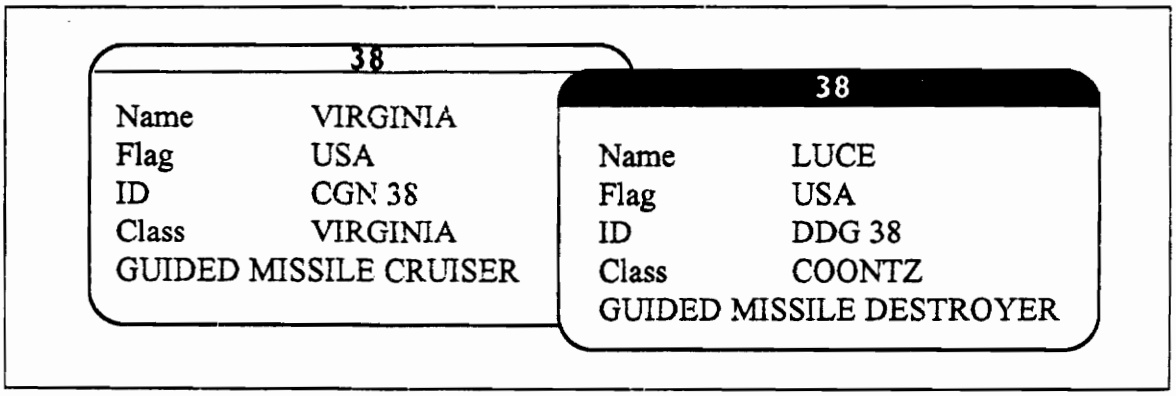


Figure 4.4.2.4 Reply to inquiry "38".

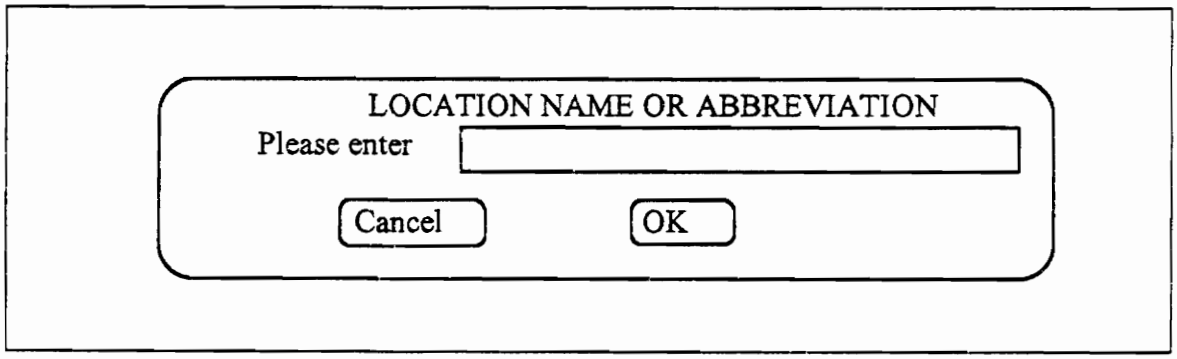


Figure 4.4.2.5 Inquiry box for location database.

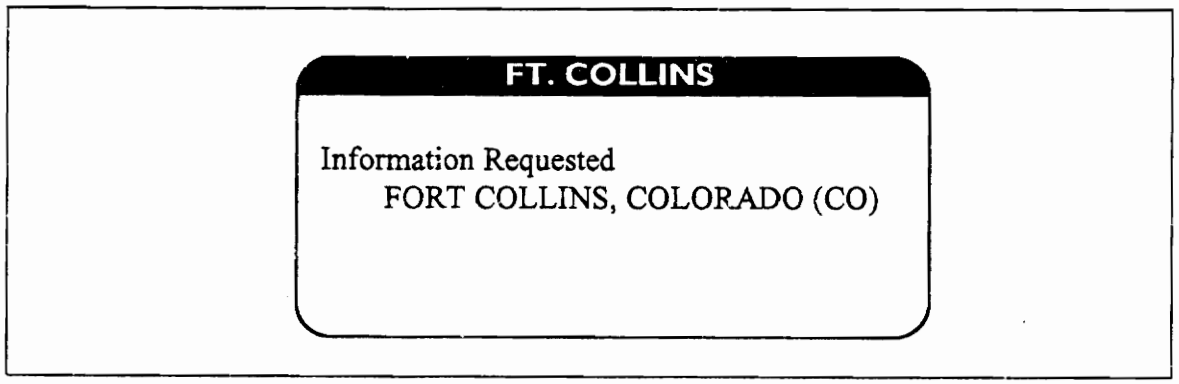


Figure 4.4.2.6 Reply to inquiry "FT. COLLINS".
A failed inquiry results in a reply of "NOT FOUND IN DATABASE" under "Information Requested".

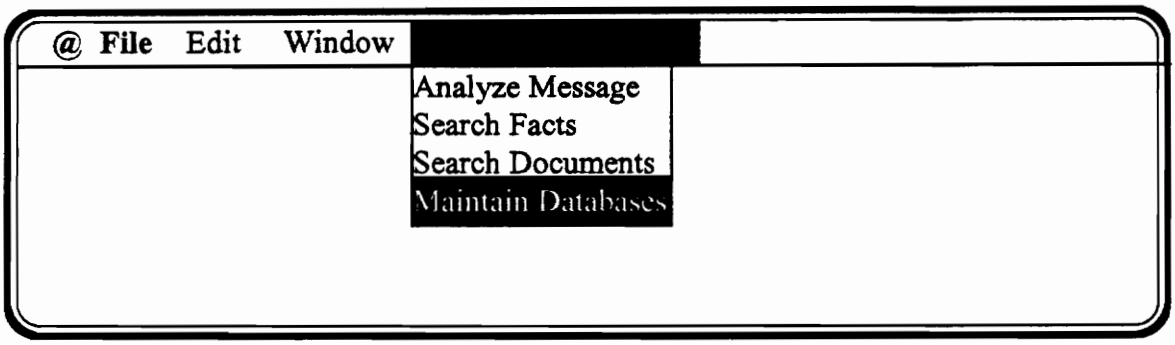


Figure 4.4.2.7 Selection of "Maintain Databases".

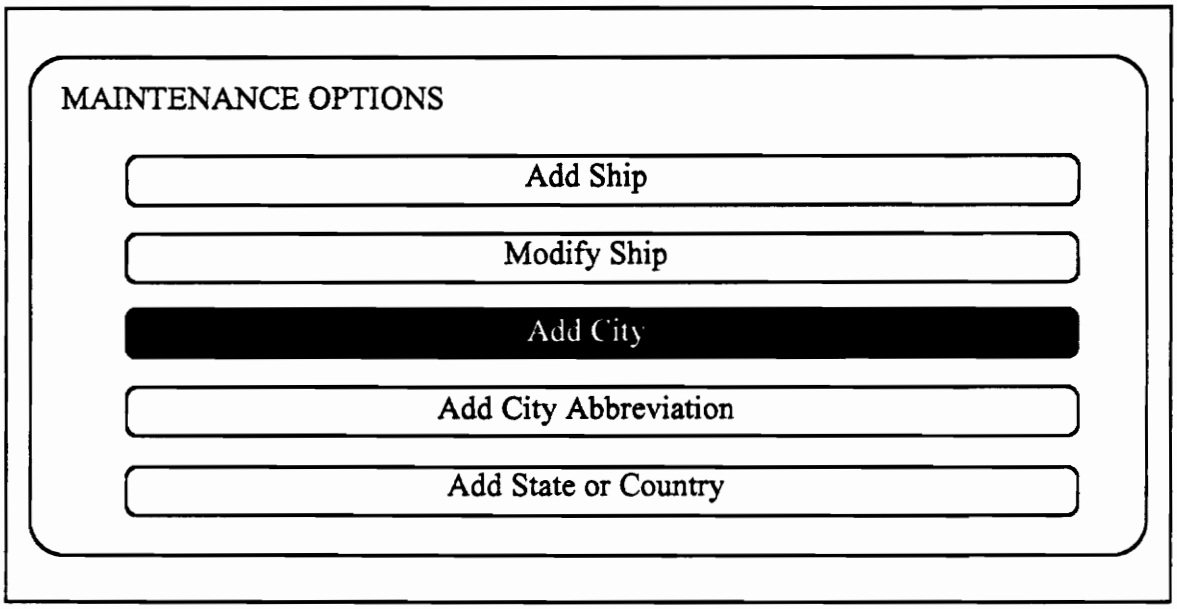


Figure 4.4.2.8 List of maintenance options, with "Add City" selected.

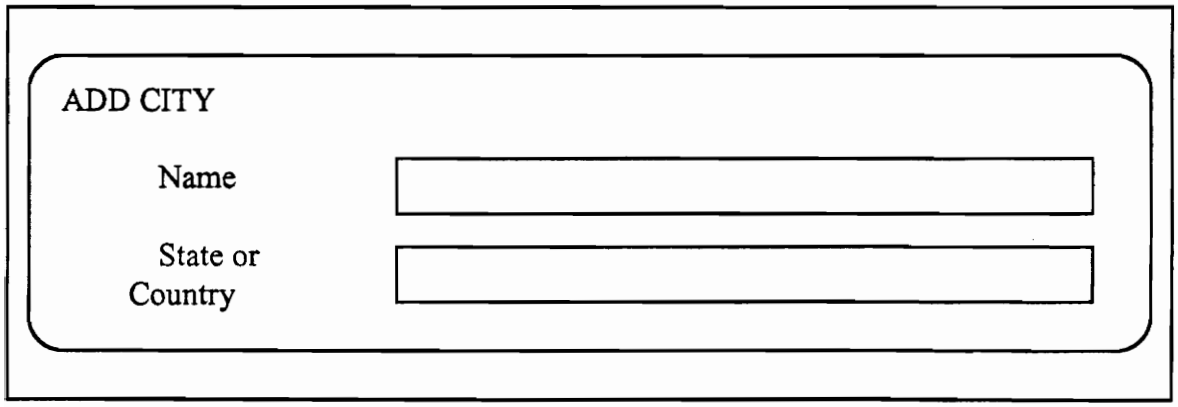


Figure 4.4.2.9 Window with edit fields to add a city to the database.

Database maintenance has several screens. First, the user selects an option, "Maintain Databases", illustrated in Figure 4.4.2.7. A list of options appears, including "Add Ship," "Modify Ship," "Add City," etc., as in Figure 4.4.2.8. Selection of an option brings up a screen with fields that can be edited for name, state, etc., as in Figure 4.4.2.9.

4.4.3 Message Retrieval Request

The request for message retrieval is made through a single screen. This subsystem received the smallest effort since the specifications did not give much detail. After the main menu selection, a single window is presented to the user. Figure 4.4.3.1 shows the format of that window. Every item matching any query item is retrieved, e.g., all documents with the date (or partial date) and all documents with the keywords filled in.

Each document is ranked according to how many criteria it fulfills. One possible criterion is the name of a city, another is a state or country abbreviation, and so on through the various indexes created during the message analysis. One "point" is given for each of these criteria fulfilled. One "point" is given for each of the keywords found in the keyword index. The documents are weighted according to how many "points" are awarded, and presented in decreasing order by the total weight, as illustrated in Figure 4.4.3.2. (Documents with zero weight are not included on the list.) This ranking method was used because it was easy to implement and adequate for the proof of concept, not because it is necessarily the best. It is likely that another method, or at least a more sophisticated weighting scheme, would prove to be better for large collections of messages.

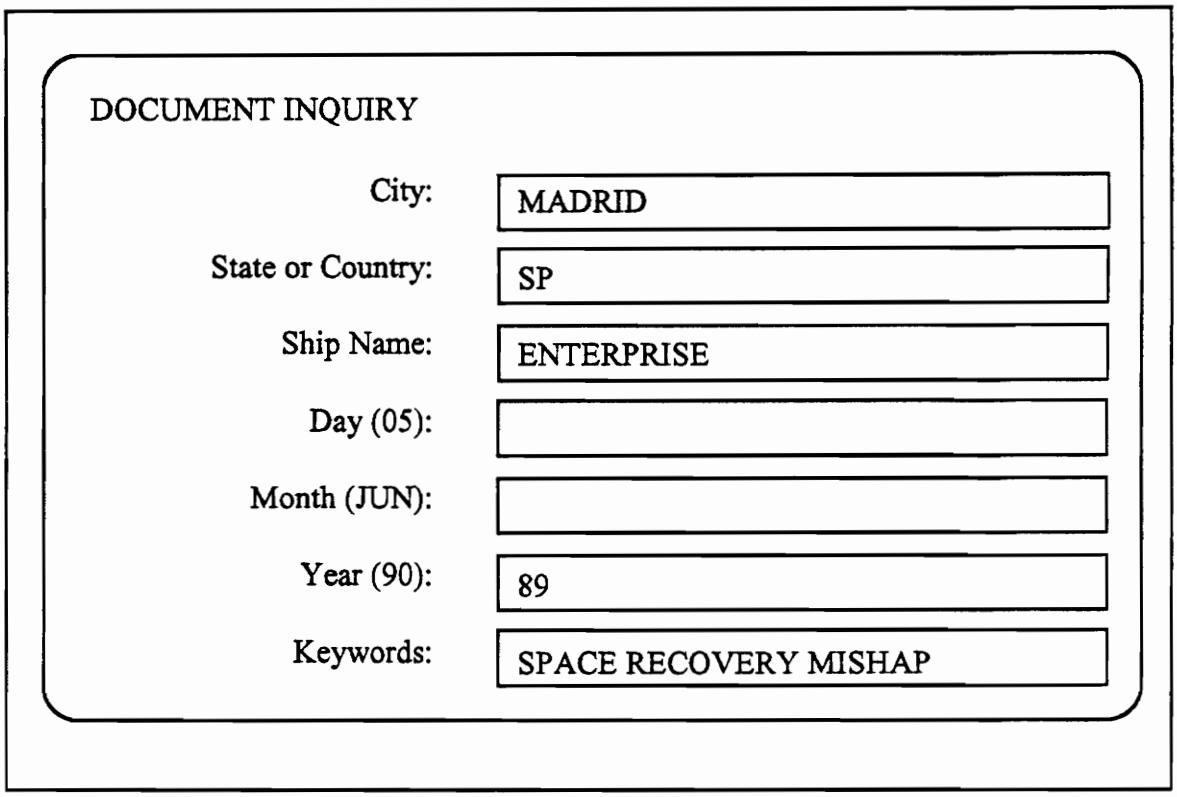


Figure 4.4.3.1 Document inquiry window with example of inquiry.

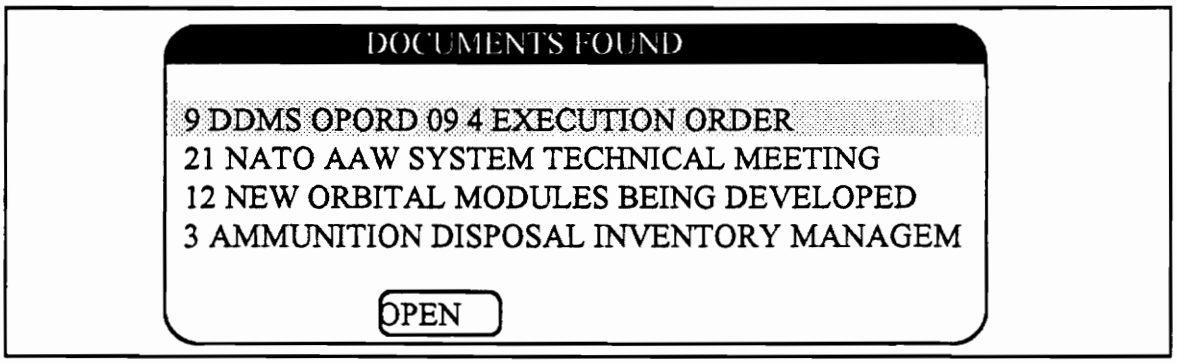


Figure 4.4.3.2 List of retrieved documents, sorted by number of criteria matched (not shown). Number shown is document reference number. Shaded subject line has been marked for viewing document.

4.5 Implementation Results

Message analysis was successful on all of the 22 sample messages placed in data files, in addition to several fictitious messages that were modified in various ways to "stress test" the NMA. The sample messages did not include all of the messages furnished, since the messages were not provided in machine-readable form. However, the system was designed to recognize all geographic locations and ship names in the world-knowledge database, and it succeeded in analyzing all the sample messages and test messages. There are some forms of date that were not present in those messages; the date expert was not designed to extract those forms, as noted in section 4.3.3.

Table 4.5 shows the number of lines of code in each module, including the number of lines in each of the fact bases. The suffix "exprt" or "expr2" denotes the main module of an expert. The suffix "util" indicates a file of utility routines, most of which were used by more than one predicate within the expert, and often in more than one of the source files of the expert. In general, source code files were limited to about 300 lines, maximum, so that modules with the suffix "2" are a continuation of the main module. (The four files exceeding 350 lines did not neatly fall into approximate halves by any obvious criterion.) This practice reduced compilation times during development.

The analysis of most messages took from three to five minutes. The messages ranged from half a page to a full page, except for one message that was a page and a half. The speed was partly due to the fact that the code was written in Prolog, which is not noted for its great speed. It was also partly due to the fact that the various computers in the system were nodes on a

Table 4.5: Lines of code in each module.
(Continued on next page)

Module Name / Source files	Lines of code	Purpose
body		separate body of message from
body.nl	80	header & trailer.
coder		initiate session
coder.nl	19	
catgrizr		determine whether message is in
catgrizr.nl	211	JINTACCS format
datexprt		extract date and time from
datexprt.nl	348	tokenized list
dateutil.nl	316	
timexprt.nl	432	
timeutil.nl	114	
freeform		search for clues in free-format
freeform.nl	267	message portions
geoexprt		extract "geographic" items from
geoexprt.nl	380	tokenized list: locations, ships
geoexpr2.nl	269	
geoutil.nl	28	
header		extract message source,
header.nl	309	destination(s), date, etc. from
header2.nl	315	message header
inquiry		handle user inquiries for message
inquiry.nl	326	retrieval
inquiry2.nl	176	
jintac		analyze JINTACCS messages
jintac.nl	78	standard types of lines
mlexprt.nl	130	multiple-line data

Table 4.5, continued
Total lines of code appear at bottom

Module Name / Source files	Lines of code	Purpose
searchdb		search LEND databases for messages
searchdb.nl	146	having date, location or ship name
srchgeog.nl	312	
srchutil.nl	177	
strategist		rule bases for strategist; linked
anal_rules.nl	96	with existing CODER object files
quest_rules.nl	47	
subject		
subject.nl	117	
toplevel		dialogue manager; initializes
toplevel.nl	306	analyses, searches, database
toplev12.nl	266	functions
topanswr.nl	428	
(other)		utilities linked to many modules
assertdb.nl	80	
utility.nl	385	
(Prolog fact bases)		world-knowledge for analysis
cluefile.nl	141	consulted at start of analysis
dateinfo.nl	97	consulted at start of analysis
geobase.nl	898	transferred to LEND databases
shipdata.nl	116	transferred to LEND databases
TOTAL LINES OF SOURCE CODE:	6158	
TOTAL LINES OF PROLOG FACTS:	1252	
TOTAL LINES OF PROLOG CODE:	7410	

campus-wide ethernet. Access to the geographic database, for example, required sending a message over the network and waiting for a reply. Every word and phrase had to be examined by this expert, as noted in section 4.3.4.

The retrieval of individual facts and of messages generally had a response time of less than two seconds. The ranking of a collection of retrieved messages in response to a query generally took from ten to fifteen seconds.

4.6 Prototypes of Further Work

Several individual projects were done by graduate students to illustrate how the NMA could be expanded and enhanced.

First, a menuing system was prototyped by Ajit Naidu on a DECstation using X Windows rather than the tool kit on the Macintosh. The choices on the menus mimicked those appearing on the Macintosh, although no attempt was made to produce a fully functional interface.

Second, a system was written by Rahesh Ursekar to display maps of various areas on the globe upon request, so that an area could be viewed for clarification if an unfamiliar geographic name were encountered in a message. This system was written to run on a Macintosh under A/UX. It was not made accessible to the CODER system due to time constraints. If this system were added to the NMA, retrieval could be initiated by marking the name of a geographic location displayed within a retrieved message, followed by retrieval of the map data from a database. Latitude and longitude information would have to be added to the database of cities and countries.

Third, silhouettes of several ships were scanned and stored in raster format on an MS-DOS machine. A large number of facts on these ten classes of ships were entered into a Prolog database. A small expert system for ship identification was written as an AI class project [BARN89] for a retrieval mechanism. Only one silhouette is stored for each class of ships, but the number of the individual ship is printed on the silhouette when it is displayed on the upper half of the screen. The detailed information concerning crew, armament, power sources, etc. is displayed on the lower half of the screen. This was also done as a stand-alone project, to demonstrate the further use of AI techniques in a military environment. If this capability were added to the NMA, retrieval could consist of highlighting the name of a ship mentioned in the message, followed by a database lookup and display of the silhouette and textual data, since the LEND database is capable of handling many types of data.

Chapter 5: Production of the Videotape

Rather than delivery of software to the Navy, the research grant required delivery of a videotape and a final demonstration of the software to representatives from the Naval Surface Warfare Center. After the demonstration, the work on the videotape was begun.

5.1 Equipment

In addition to the standard studio equipment, there were several items available because of other projects. A test version of the DVI® system from Intel was set up nearby. This included a 386 microcomputer running under DOS, as well as special boards for audio and video capture and editing. A monochrome monitor served as the system monitor, with a high-resolution RGB monitor for video editing. The hardware is capable of capturing, compressing, decompressing , and presenting full-motion video, although those capabilities were not needed for this project.

5.2 Software

There are four primary pieces of software used on the DVI system for this project. First, there is an image-capture and compression facility which is used to digitize images ("frames") in the standard U.S. television format [DVIT90]. Second, there is an audio capture and compression facility that can receive input from a microphone, CD player or tape deck [DVIT90]. Third, there are image editing facilities, of which LUMENA™ [LUME89] was used most. Fourth, there is a script playback and video editing facility called MEDIAscript™ [LUTH90] that can access audio and video files, play audio while displaying pictures, etc.

There was no facility for capturing and editing full-motion, full-screen video in the quality and quantity required for several minutes of production. Therefore, all video production on the DVI system was with a series of still frames. The introduction and conclusion by the narrator were videotaped, as well as four short sequences showing a prototype of the NMA menus under X Windows. Titling, music and final production were done in the studio at Virginia Tech.

5.3 Transfer of Screen Images to Disk

There were three major attempts made to transfer the Macintosh screen images to digitized files. (There is a board available that will capture the screen directly, but A/UX would not boot with the board in place. Also, the resolution of standard TV is so low that many of the characters would be unreadable.)

The first attempt was to photograph the screens and make slides, which could then be transferred to videotape or to disk. This would have the advantage of avoiding the severe flicker of the image due to the differing scan rates of television and the Macintosh screen. This was attempted, but there were two main problems: the resolution became worse as more equipment and processing came between the screens and the final image, and the light levels of the image became brighter in the center and dimmer at the edges.

The second attempt was to capture the screen image on videotape and then play back the image for digitizing. The resolution of standard U.S. television and the additional loss of resolution on VHS videotape defeated this attempt also.

The third attempt consisted of using the video camera and stringing a video cable directly to the DVI lab to achieve direct input and video capture. This attempt was highly successful, and the screens are readable when played back on an ordinary color television set. This technique is probably better under the circumstances than direct capture of video on the Macintosh, since the camera was used to take close-up pictures of the significant parts of the screen, making the characters larger and hence more readable. The flicker due to different scan rates proved to be a minor annoyance, since the enlarged image was of uniform brightness whenever the brightest part of the image was out of the picture. Generally, only two or three frames needed to be examined to find one that was satisfactory. Forty-nine frames were selected for the video production. Little or no video editing needed to be done with these images.

5.4 Chart and Diagram Production

The LUMENA™ [LUME89] program was used to produce all the charts and diagrams except for a few charts at the end. LUMENA™ uses both the monochrome and RGB monitors in conjunction with the mouse. The mouse cursor moves freely between the screens.

The LUMENA™ package includes most of the standard paint and draw functions, including shapes, lines, curves, fonts, shading and a palette of 4096 colors. Animation is available, but was not used here. (Some animation was done using the MEDIAscript™ facility described in the next paragraph.) 29 charts and diagrams of [BARN90b] were done with LUMENA™.

The MEDIAscript™ package [LUTH90] is more limited in the facilities for drawing and painting, with fewer fonts than LUMENA™ and fewer colors. (Although thousands of colors can be displayed by the program from image files, a relatively small palette is available within the MEDIAscript™ facilities.) On the other hand, it is possible to make diagrams that dynamically change according to a sequence of commands in the script file. For one of the screens, for example, it was desirable to show several computers with communication links. It was possible to make the communication links in the diagram blink by placing commands in the script file to redraw them in different colors. On the same chart, some of the labeling was made to appear in sequence along with the narration. LUMENA™ charts must be loaded and displayed sequentially, which is much slower, and would not proceed as fast as the narration at these critical points. A total of ten charts and diagrams were done with MEDIAscript™ .

Three photographs of computers were transferred from slides to videotape.

5.5 Audio Production

Audio production was done by reading from a script and recording on reel-to-reel tape. The recorder was furnished by the Virginia Tech Audio-Visual Services. The tape was then transferred to one large disk file (compressed) for further processing. Because of the difficulty in producing a single long segment of video coordinated with sound, the audio was broken into several segments using the audio editor furnished with the capture facility. The files were then accessed sequentially according to the final script.

Several extraneous noises were introduced during the transfers to and from disk files. These were removed with the editing facilities at the studio.

5.6 Coordinating Audio with Video

The MEDIAscript™ software provides for the creation and running of a script file to present both audio and video [LUTH90]. Through the use of coprocessors, the audio can be started and played from one disk file while pictures are loaded and displayed from another disk file. One picture is displayed on the screen while another is being loaded from disk. (There are also facilities for user input and branching according to the response. These facilities were not needed for the current project.)

Coordination begins with playing the audio and finding the approximate time when each still frame should appear. The script contains timing information for the pauses between pictures. During the first few runs of the script, the timing is made more precise at the beginning; that is, the pictures are made to appear at precise times during the narrative. As more runs are made, the times are adjusted at points farther into the script. Unfortunately, there is not a command that a picture should appear exactly n seconds after the start, but only one indicating how many seconds should elapse between pictures. Thus the procedure becomes $O(n^2)$ as measured from the beginning of the script, to adjust the timing n seconds into the script. Therefore, the audio was broken into four files and four scripts were produced. In the final production, all four scripts were concatenated.

Table 5.6 gives an outline of the video in its final form.

Table 5.6 Outline of Video

The Naval Message Analyzer Video	
I. Introduction	47 seconds narration 14 slides
II. CODER	
a. Overview	3 slides
b. Blackboard systems	11 slides
III. NMA	
a. Start-up	1 slide 3 screens
b. Message Analysis	3 slides 2 photographs
c. Database Inquiry	10 screens
d. Database Maintenance	6 screens
e. Document Search	5 screens
IV. AIList/CED	
a. Start-up	1 screen
b. Vector Query of AIList	4 screens
c. Dictionary Inquiry	4 screens
d. Query Refinement	10 screens
V. Potential Enhancements	
a. Introduction	1 slide
b. X Windows/Motif	28 seconds full motion
c. E-Mail and Speedups	6 slides
d. Other Experts	7 slides
VI. Conclusion	16 seconds narration 7 seconds full motion
VII. Credits	

Chapter 6: Future Work

There are many individual projects that could be done to extend or adapt the Naval Message Analyzer. Three such prototypes are reviewed in Section 4.6. The X Windows menuing prototype works toward a solution to the interface problem described in Section 6.9. The presentation of maps is mentioned in Section 6.2, and the display of ship silhouettes and other ship information is described in Section 6.1.

6.1 Knowledge of Ships

A database consisting of silhouettes and detailed information on range, speed, engines, armaments, etc. from *Jane's Fighting Ships* [JANE89] would be an enhancement to the presently stored world knowledge. Vital statistics on other ships could be made available along with the standard silhouette of the ship for visual confirmation. The LEND database is an ideal tool for such data storage, since it has the capability to store data in frames as well as entire documents (narrative descriptions) and other objects (digitized images). A prototype is described in Section 4.6 and in [BARN89].

6.2 Geographical Knowledge

The geographical databases could be expanded to include much more detailed information such as exact city coordinates, populations, terrain and major industries. With the use of graphical workstations and the X Window System, world, regional and local maps could be displayed upon request during retrieval, along with other information such as present position of the user's ship.

Another enhancement would be to parse items with geographical words, even though they are not in the geographical databases. For example, "China Lake" or "Lake Erie" might result in a query to a human operator and, if appropriate, could be added to one of the databases.

6.3 Dates

There are several forms of the date that the date expert cannot handle, for example "January of next year", "last Tuesday" or "0400 this morning". Handling such instances can be added to the capabilities of the system, simply by using the message date, a day-of-the-week algorithm, and perhaps some other specialized routines to handle seldom-occurring patterns such as "ten days hence".

Due to the rapid changing of the database itself, a better method of expressing temporal dependency for both analysis and retrieval might be useful [NAVA89]. Additions and corrections sent in later messages might also be tracked by the system ("the frame problem") [DEAN89].

6.4 Subject Indexing

A list of standard keywords and subject headings is used by the Navy. This list could be used by the subject expert for automatic indexing and retrieval, with the more general indexing by words, locations and dates as a second method. Retrieval by subject might present a menu of subjects to the user, for example.

If the message is a standard report as indicated by the subject, and the message consists of filled-in items on a form as in Figure 4.2.2, the user could be presented with the filled-in form with the responses highlighted.

6.5 JINTACCS Expert

The JINTACCS expert can easily be extended to record all information in a database. Every field within a message is labeled, and whenever information appears in columns, the forward slash is used as the delimiter between columns on each line. If the information were to be recorded in a Prolog fact base, even the names of the functors could be taken directly from the column headings; e.g., in Figure 4.1.2.1, one such fact might be

```
tgtdata(msgid('INTREP','MAGISMS','0001'),
                de('01'),tgttyp('MDMTK'),
                eqmt('T-62'),eqmod('-'),qty(10)).
```

where the message identifier ("msgid") is taken from the header and previous information in the body.

The parsing of JINTACCS messages would not be difficult, but it may be that Prolog is not the correct tool for further refinement of this expert; the number of patterns in Prolog would be very large. A much simpler algorithm could be developed in a procedural language such

as C. The Prolog JINTACCS expert could then post an appropriate message on the blackboard and the C JINTACCS expert would then parse the columnar data. All free-text fields such as 'RMKS' (remarks), 'AMPN' (amplification) or 'NARR' (narration) would, of course, be submitted to the free-format expert for parsing.

6.6 Messages in Other Formats

Messages from other sources were included in the collection of sample messages. These included messages from the weather service (NOAA) in particular. There were several sets of data, e.g., wind speed and direction predicted at various times, presented in tabular form. An expert to read such tables would be useful. Experts could also be written for analyzing messages from other military branches, other government agencies, and foreign governments.

The capability of distinguishing additional formats from JINTACCS and each other could be added. No other formats were included in the sample, but analysis could proceed along the same lines as the current project.

Because of the specialized nature of military messages, the morphological analyzer could be extended (or a separate module built) specifically for abbreviations. Some abbreviations could be analyzed by an algorithm, e.g., "CINCPACFLT" (Commander IN Charge PACific FLeeT), while others might require a table for interpretation. Some work has been done in this area, for abbreviations of single words, instead of military abbreviations [MEAN88].

6.7 Retrieval Subsystem

The retrieval subsystem needs further study and probable modification. The studies could take several directions. First, the use of OR for retrieval followed by ranking on the number of "hits" needs to be examined more closely. Previous work indicates difficulty with formulation of queries by inexperienced users, especially the confusion of AND and OR [MARC89a]. Furthermore, the order of presentation of retrieved documents appears to influence user judgments concerning relevance [EISE88].

Second, the use of fuzzy requests using linguistic quantifiers might also prove useful in some situations [KACP89], as well as fuzzy indexing [BOSC89] or weighting of query terms [SANC89]. Fuzzy databases have already been used in construction of military budgets for command, control and communications [DOCK89].

There are other techniques which would be very useful to the user for retrieval, such as the use of "searchonyms," that is, synonyms in the context of the present search [ATTA77]. A hypertext-style browsing subsystem might lead to improved retrieval.

6.8 Message Understanding

A script-based system such as FRUMP [DEJO82] could be implemented to summarize messages. Scripts could be developed for the areas deemed highest priority by the Navy. Message routing and retrieval might also be refined by this technique.

6.9 Interface

The interface needs to be more standard and portable. One possibility is to use the X Window System as a standard. This would allow easier construction of windows and would run on a wider variety of machines.

6.10 People

A method of recognizing personal references through the use of titles would not be difficult to implement. Clues such as "MR", "MRS", "DR", "LT", "ADM", etc. could be placed in the clue file, with the associated expert "people", for example.

6.11 Maps

With graphics available to the user interface, as on the Macintosh, geographic inquiries could offer an option to present a map showing the location sought. A command could be available when a message is presented to allow the user to mark geographic locations and display a map with the location highlighted.

6.12 Security

Security is not a consideration in most data storage and retrieval environments, but it is of extreme importance in the military. Therefore, several suggestions are made here for modification of some parts of the NMA system as developed. Some of these suggestions for security techniques would be abhorrent to some academic users but necessary for the military.

The LEND database system could be modified to secure messages according to classification. Classification of messages would be included as part of the key. Encryption/decryption would be automatic, varying according to classification. Unclassified messages, for example, might not need to be encrypted at all. Furthermore, Secret and Top Secret messages might require special routines and storage on devices not accessible at all through the UNIX file system. In this case, perhaps only a pointer to the message would be available within the file system.

The user interface would also have to have some means of identifying the particular machine it is running on, and communicating the security level to the rest of the NMA system. Certain locations are secure while others are not, and people without adequate security clearances should not have opportunity to see a screen displaying classified messages. This identification would probably be made through the installation of a board plugged into the bus, or a "dingle" plugged into a printer or communications port. In the case of highest security, the user might be required to furnish his or her own hardware device along with a memorized password in addition to the usual system ID and password.

The document retrieval system should not even give a clue to the existence of classified messages that otherwise fit the user's query, if the user's clearance and the security of the location are not high enough. If there is a real need to know, an individual with adequate clearance must do the search.

6.13 Weighting of Retrieval Criteria

With a large sample of messages, the weighting of retrieval items could be varied to emphasize various items. For example, the ship name could receive greater weight than the date, or the keywords could each receive greater weight than the city. The user could have control of the weighting factors, or they could be stored in a user profile. Various similarity measures could also be tested.

Chapter 7: Conclusions

This project has demonstrated the potential of integrating artificial intelligence and information storage and retrieval methods for distributed, effective and easy-to-use analysis and searching of Navy messages. The CODER environment is good for the development of systems such as the Naval Message Analyzer. The high degree of modularity inherent in the blackboard system allows independent development of modules and subsystems. Future development of systems of this magnitude should not be undertaken by a single investigator, however, due to the size of the system and the inherent modularity.

The CODER system may or may not be adequate for implementation of real-time distributed message-analysis systems. Implemented in CODER over a campus-wide ethernet, the analysis phase appeared to be slow. Dedicated computers directly connected to each other would probably increase the speed. As shown in Figure 4.0.4, there is an information bottleneck at the LEND database server; database servers for different experts should reside on separate machines for maximum speed, each with a local copy of the appropriate database. (It would be unwise to have multiple copies of the database server during development, however, since the correction of faults in the database server and/or the data itself is an ongoing process.) In either case, multiple copies of the data should be avoided. It may also be that the blackboard itself is a bottleneck, in which case more direct calls to servers could be used in place of blackboard posting. One such example is the user interface, which also retrieves the requested documents without the use of a separate blackboard posting. Implementations of CODER on a smaller network and elimination of the bottlenecks should yield sizeable improvements. The CODER system has shown itself to be successful as a testbed for multiprocessing projects such as the NMA.

List of References

[APTE87] Apte, S. "Communication in the CODER System." Masters Thesis. Blacksburg VA: Virginia Tech Department of Computer Science, June 1987.

[ATTA77] Attar, R. and A. S. Fraenkel. "Local Feedback in Full-Text Retrieval Systems." *J. ACM* 24(3): 397-417, July 1977.

[BARN87] Barnhart, R. D. "The CODER Lexical Expert." Class Project Report. Blacksburg VA: Virginia Tech Department of Computer Science, June 1987.

[BARN88] Barnett, G. A. and G. Sigel. "The Diffusion of Computer-Assisted Legal Research Systems." *JASIS* 39(4):214-224, 1988.

[BARN89] Barnhart, R. D. "Expert System Visual Ship Identifier." Class Project. Blacksburg VA: Virginia Tech Department of Computer Science, November 1989.

[BARN90a] Barnett, J., Knight, K., Mani, I. and E. Rich. "Knowledge and Natural Language Processing." *Communications of the ACM* 33(8):50-71, August 1990.

[BARN90b] Barnhart, R. D. "The Advanced Naval Message Analyzer." Videotape Production, Virginia Tech, November 1990. Richard Barnhart as host, producer, script-writer; Edward Fox as executive producer, project director. Discussed the CODER system and its application to Naval message analysis and retrieval.

[BARR81] Barr, A. and E. Feigenbaum. *The Handbook of Artificial Intelligence, vol. 1.* Los Altos, CA: William Kaufmann, Inc., 1981.

[BARR82] Barr, A. and E. Feigenbaum. *The Handbook of Artificial Intelligence, vol. 2.* Los Altos, CA: William Kaufmann, Inc., 1981.

[BARR89] Barr, A., Cohen, P. R. and E. A. Feigenbaum. *The Handbook of Artificial Intelligence, vol. 4.* Los Altos, CA: William Kaufmann, Inc., 1981.

[BART83] Bartschi, M. and H. P. Frei. "Adapting a Data Organization to the Structure of Stored Information." In Salton, Gerard and Hans-Jochen Schneider (Eds.), *Research and Development in Information Retrieval, Proc. Berlin, May 18-20, 1982.* Berlin: Springer-Verlag, 1983, pp. 62-79.

[BATE79a] Bates, M. J. "Information Search Tactics." *Journal of the American Society for Information Science* 30(2):205-214, July 1979.

[BELK82] Belkin, N. J. et al. "Ask for Information Retrieval: Part I. Background and Theory." *Journal of Documentation* 38(2):61-71, June 1982.

[BELK83] Belkin, N. J., Seeger, T. and G. Wersig. "Distributed Expert Problem Treatment as a Model for Information System Analysis and Design." *Journal of Information Science*, 5:153-167, 1983.

[BELK84] Belkin, N. J., Hennings, R. D. and T. Seeger. "Simulation of a Distributed Expert-based Information Provision Mechanism." *Information Technology Research & Development Applications*, 3(3):122-141, 1984.

[BELK87a] Belkin, N. J., Borgman, C., Brooks, H., Bylander, T., Croft, W. B., Daniels, P., Deerwester, S., Fox, E. A., Ingwersen, P., Rada, R., Sparck-Jones, K., Thompson, R. and D. Walker. "Distributed Expert Systems: An Interdisciplinary Approach." *Information Processing and Management* 23(5):395-409, 1987.

[BELK87b] Belkin, N. J. and W. B. Croft. "Retrieval Techniques." *Annual Review of Information Science and Technology* 22:109-145, 1987.

[BELK87c] Belkin, N. J. and H. M. Brooks. "Knowledge Elicitation Using Discourse Analysis." *International Journal of Man-Machine Studies* 27:127-144: 1987.

[BENW86] Benwell, N. J. *Data Preparation Techniques*. London: Advance Publications Ltd., 1976.

[BERK91] Berk, E. and J. Devlin, eds. *Hypertext/Hypermedia Handbook*. New York: McGraw-Hill, 1991.

[BLAI85] Blair, D. C. and M. E. Maron. "An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System." *Communications of the ACM* 28(3):289-299, March 1985.

[BOBR85] Bobrow, D. G. "If Prolog is the Answer, What is the Question? or What it Takes to Support AI Programming Paradigms." *IEEE Trans. on Software Engineering* 11(11): 1401-1408, November 1985.

[BOLC80a] Bolc, L., ed. *Representation and Processing of Natural Language*. Munich: Carl Hanser Verlag, 1980.

[BOLC80b] Bolc, L., ed. *Natural Language Based Computer Systems*. Munich: Carl Hanser Verlag, 1980.

[BOOK80] Bookstein, A. "Fuzzy Requests: An Approach to Weighted Boolean Searches." *J. Am. Soc. Inf. Sci.* 31(4):240-247, July 1980.

[BORG84] Borgman, C. L. "Psychological Research in Human-Computer Interaction." *ARIST* 19:33-64, 1984.

[BORG85] Borgman, C. L. "Designing an Information Retrieval Interface Based on User Characteristics." *Proceedings of the 8th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval (Montreal: June 1985) ACM,* 139-146, 1985.

[BOSC89] Bosc, P. and M. Galibourg. "Indexing Principles for a Fuzzy Data Base." *Information Systems* 14(6): 493-499, 1989.

[BOYN85] Boyne, W. J. and H. Otane. "Direct Document Capture and Full Text Indexing: An Introduction to the National Air and Space Museum System." *Library Hi Tech* 8:7-14, 1985.

[BRAC88] Brach, R. J. and D. L. McGuiness. "Knowledge Representation, Connectionism, and Conceptual Retrieval." *Proceedings of the 11th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval (Grenoble, France, June 13-15, 1988) ACM,* 161-174, 1988.

[BRAT86] Bratko, I. *Prolog Programming for Artificial Intelligence.* Reading MA: Addison-Wesley, 1986.

[BROO87] Brooks, H. M. "The Functions of an Information System: The MONSTRAT Model." Paper presented at the Workshop on Distributed Expert-Based Information Systems. School of Communication, Information and Library Studies, Rutgers University, March 1987.

[BUCK85] Buckley, C. "Implementation of the SMART Information Retrieval System." TR 85-686, Cornell Univ., Dept. of Comp. Sci, May 1985.

[BUSH45] Bush, V. "As We May Think." *Atlantic Monthly* 176:101-108, July 1945.

[BUSH55] Bush, V. "Communications--Where Do We Go from Here?" *Mechanical Engineering* 77(4): 302-304, 1955.

[CASE85] Case, D. "The Personal Computer: Missing Link to the Electronic Journal?" *JASIS* 36(5): 309-313, 1985.

[CHAN86] Chandrasekaran, B. "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design." *IEEE Expert* 1(3):23-30, Fall, 1986.

[CHAR86] Charniak, E. And D. McDermott. *Introduction to Artificial Intelligence.* Reading MA: Addison-Wesley, 1985.

[CHEN84] Chen, C. and P. Hernan, ed. *Numeric Databases.* Norwood NJ: Norwood Publishing Corp., 1984.

[CHEN90] Chen, Q-F. "An Object-Oriented Network Database System for Information Retrieval Applications. Dissertation Proposal, Virginia Tech Dept. of Computer Science, August, 1990.

[CHER87a] Chernicoff, S. *Macintosh Revealed, Volume One: Unlocking the Toolbox, Second Edition.* Indianapolis, IN: Hayden Books, 1987.

[CHER87b] Chernicoff, S. *Macintosh Revealed, Volume Two: Programming with the Toolbox, Second Edition.* Indianapolis, IN: Hayden Books, 1987.

[CHIA84] Chiaramella, Y. and B. Defude. "Prototype of an Intelligent System for Information Retrieval: IOTA." *Information Processing & Management* 23(4): 285-303, 1987.

[CLOC84] Clocksin, W. F. and C. S. Mellish. *Programming in Prolog.* New York, NY: Springer-Verlag, 1984.

[COBB87] Cobb, S. *Using Reflex: The Database Manager.* Berkeley CA: Osborne McGraw-Hill, 1989.

[CODD78] Codd, E. F. "How About Recently? (English Dialog with Relational Data Bases Using Rendezvous Version I)." in Schneiderman, B., ed. *Databases: Improving Usability and Responsiveness.* New York: Academic Press, 1978.

[CODD90] Codd, E. F. *The Relational Model for Database Management: Version 2.* Reading MA: Addison-Wesley Publishing Co., 1990.

[COHE82] Cohen, P. R. and E. Feigenbaum. *The Handbook of Artificial Intelligence, vol. 3.* Los Altos, CA: William Kaufmann, Inc., 1981.

[COOM90] Coombs, J. H. "Hypertext, Full Text and Automatic Linking." *Proceedings of the 13th International Conference on Research and Development in Information Retrieval, Brussels, Belgium, 5-7 September 1990.* New York: The Association for Computing Machinery, 1990.

[CORK87] Corkill, D. D., Gallagher, K. Q. and P. M. Johnson. "Achieving Flexibility, Efficiency and Generality in Blackboard Architectures." *Proceedings of the Sixth National Conference on Artificial Intelligence (Seattle WA: July 13-17, 1987)*, AAAI, pp. 18-23, 1987.

[CRAW86] Crawford, R. G. and H. S. Becker. "A Novice User's Interface to Information Retrieval Systems." *Information Processing & Mangagement* 22(4) pp. 287-298, 1986.

[CROF85] Croft, W. B. and T. J. Parenty. "A Comparison of a Network Structure and a Database System Used for Document Retrieval." *Information Systems* 10:4 (1985), pp. 377-390.

[CROF87] Croft, W. B. and R. Thompson. "I^{3}R: A New Approach to the Design of Document Retrieval Systems." *JASIS* 38(6):389-404, November 1987.

[CROF88] Croft, W. B., Lucin, T. J. and P. R. Cohen. "Retrieving Documents by Plausible Inference: A Preliminary Study." *Proceedings of the 11th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval (Grenoble, France, June 13-15, 1988) ACM,* pp. 481-494, 1988

[DATE86] Date, C. J. *An Introduction to Database Systems, Fourth Edition.* Reading MA: Addison-Wesley Publishing Co., 1986.

[DEAN89] Dean, T. "Using Temporal Hierarchies to Efficiently Maintain Large Temporal Databases" *Journal of the ACM* 36(4): 687-718, Oct. 1989.

[DEER85] Deering, M. F. "Hardware and Software Architectures for Efficient AI." *Byte* 10(4):193-198, 1985.

[DEIT85] Deitel, H. M. and B. Deitel. *Computers and Data Processing.* New York: Academic Press, 1985.

[DEJO82] DeJong, G. "An Overview of the FRUMP System." In Wendy G. Lehnert and Martin H. Ringle (Eds.), *Strategies for Natural Language Processing.* Hillsdale NJ: Lawrence Erlbaum Assoc., 1982, pp. 149-176.

[DESA86] Desai, B. C., Goyal, P. and F. Sadri. "A Data Model for Use with Formatted and Textual Data." *JASIS* 37(3):158-165, 1986.

[DIET84] Dietschmann, H. J., ed. *Representation and Exchange of Knowledge as a Basis of Information Processes.* New York: North-Holland, 1984.

[DIGI89a] *Digital Video Interactive Technology: Pro 750™ ADP Computer Installation Guide.* Santa Clara, CA: Intel Corp., 1989.

[DIGI89b] *Digital Video Interactive Technology: Pro 750™ ADP System Setup Guide.* Santa Clara, CA: Intel Corp., 1989.

[DILL90] Dillon, A. and C. McKnight. "Towards a Classification of Text Types: A Repertory Approach." *International Journal of Man-Machine Studies* 33: 623-636, 1990.

[DOCK89] Dockery, J. T. and E. Murray. "Fuzzy Linguistic Data Bases: An Application." *Information Systems* 14(6): 443-453, 1989.

[DOYL85] Doyle, J. "Expert Systems and the 'Myth' of Symbolic Reasoning." *IEEE Transactions in Software Engineering* 11(11): 1386-1390, 1985.

[DUNL87] Dunlop, N. *Working with Q&A: Practical Techniques in Database Design.* Berkeley CA: Osborne McGraw-Hill, 1989.

[DVIT90] *DVI™ Technology: ActionMedia™ 750 Production Tool Reference.* Santa Clara, CA: Intel Corp. 1989.

[EISE88] Eisenberg, M. and C. Barry. "Order Effects: A Study of the Possible Influence of Presentation on User Judgments of Document Relevance." *JASIS* 39(5): 293-300, 1988.

[ERIC86] Erickson, J. and N. Baran. *Using R:BASE 5000.* Berkeley CA: Osborne McGraw-Hill, 1989.

[ERMA80] Erman, D. L., Hayes-Roth, F., Lesser, V. R. and D. R. Reddy. "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty." *ACM Comp. Surveys* 12(2): 213-253, June 1980.

[FALO85] Faloutsos, C. "Access Methods for Text." *ACM Comp. Surveys* 17(1):49-74, March 1985.

[FIKE85] Fikes, R. and T. Kehler. "The Role of Frame-Based Representation in Reasoning." *Communications of the ACM* 28(9):904-920, September 1985.

[FIND79] Findler, N., ed. *Associative Networks.* New York: Academic Press, 1979.

[FORE85] Forester, T., Ed. *The Information Technology Revolution.* Cambridge MA: The MIT Press, 1985.

[FOXE80] Fox, E. A. "Lexical Relations: Enhancing Effectiveness of Information Retrieval Systems." *ACM SIGIR Forum* 15(3):5-36, Winter 1980.

[FOXE81] Fox, E. A. "Implementing SMART for Minicomputers Via Relational Processing with Abstract Data Types." In *Joint Proc. of SIGSMALL Symp. on Small Systems and SIGMOD Workshop on Small Data Base Systems, ACM SIGSMALL Newsletter* 7(2):119-129, October 1981.

[FOXE83a] Fox, E. A. "Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts." TR 83-561, Cornell University, Department of Computer Science, September 1983.

[FOXE83b] Fox, E. A. "Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types." Dissertation, Cornell University, University Microfilms Int., Ann Arbor MI, August 1983.

[FOXE83c] Fox, E. A. "Some Considerations for Implementing the SMART Information Retrieval System under UNIX." TR 83-560, Cornell University, Department of Computer Science, September 1983.

[FOXE85a] Fox, E. A. "Combining Information in an Extended Automatic Information Retrieval System for Agriculture." In El-Hadidy, B. and E. E. Horne (Eds.), *The Infrastructure of an Information Society.* North Holland, 1984, pp. 449-466.

[FOXE85b] Fox, E. A. "Composite Document Extended Retrieval: An Overview." In *Res. & Dev. in Inf. Ret., Eighth Annual Int. ACM SIGIR Conf., Montreal, 42-53, June 1985.*

[FOXE86a] Fox, E. A. and R. K. France. "A Knowledge-Based System for Composite Document Analysis and Retrieval: Design Issues in the CODER Project." TR-86-6, Virginia Tech Dept. of Comp. Sci., March 1986.

[FOXE86b] Fox, E. A. and R. K. France. "Architecture of a Distributed Expert System for Composite Document Entry, Analysis, Representation and Retrieval." P*roceedings Third Annual USC Comp. Sci. Symp.; Knowledge-Based Systems: Theory and Applications, Columbia, SC: March 31-April 1, 1986.*

[FOXE86c] Fox, E. A. and R. K. France. "Architecture of an Object-Oriented System for Composite Document Analysis, Representation and Retrieval." TR-86-10, Virginia Tech Dept. of Comp. Sci., April 1986.

[FOXE86d] Fox, E. A. and S. Sharat. "A Comparison of Two Methods for Soft Boolean Operator Interpretation in Information Retrieval." TR-86-1, Virginia Tech Dept. of Comp. Sci., January 1986.

[FOXE86e] Fox, E. A. "Improved Retrieval Using a Relational Thesaurus Expansion of Boolean Logic Queries." In Martha Evens (Ed.), *Relational Models of the Lexicon*. New York: Cambridge Univ. Press, 1986, pp. 199-210.

[FOXE86f] Fox, E. A. "Information Retrieval: Research into New Capabilities." In Suzanne Ropiequet and Steve Lambert (Eds.) *CD-ROM: The New Papyrus.* Redmond WA: Microsoft Press, 1986, pp. 143-174.

[FOXE87a] Fox, E. A. "Development of the CODER System: A Testbed for Artificial Intelligence Methods in Information Retrieval." *Information Processing & Management* 23(4):341-366, 1987.

[FOXE87b] Fox, E. A. and Q-F. Chen. "Text Analysis in the CODER system." *Proceedings Fourth Annual USC Computer Science Symposium: Language and Data in Systems, Columbia SC: April 8, 1987.* pp. 7-14.

[FOXE87c] Fox, E. A. and R. K. France. "Architecture of an Expert System for Composite Document Analysis, Representation and Retrieval." *International Journal of Approximate Reasoning* 1(2):151-175, 1987.

[FOXE88] Fox, E. A., Weaver, M. T., Chen, Q-F. and R. K. France. "Implementing a Distributed Expert-Based Information Retrieval System." *Proceedings RIAO (Recherche d'Informations Assistee par Ordinateur) 88: User-Oriented Text and Image Handling March 21-24, 1988, Cambridge MA*, pp. 708-726.

[FOXE91a] Fox, E. A., Koushik, M. P., Chen, Q-F. and R. K. France. "Integrated Access to a Large Medical Literature Database." TR 91-15, Virginia Tech, Dept. of Comp. Sci, May 1991.

[FOXE91b] Fox, E. A., Chen, Q-F. and R. K. France. "Integrating Search and Retrieval with Hypertext." In Berk, E. and J. Devlin, (Eds.) *Hypertext/Hypermedia Handbook.* New York: McGraw-Hill, 1991, pp. 329-355.

[FRAN86a] France, R. K. "An Artificial Intelligence Environment for Information Retrieval." Masters Thesis. Blacksburg VA: Virginia Tech Department of Computer Science, June 1986.

[FRAN86b] France, R. K. and E. A. Fox. "Knowledge Structures for Information Retrieval: Representation in the CODER Project." *Proceedings IEEE Expert Systems in Government Symposium, McLean VA: 20-24 October 1986.* IEEE, 1986, pp. 135-141.

[FREI83] Frei, H. P. and J. F. Jauslin. "Graphical Presentation of Information and Services: A User Oriented Interface." *Inf. Tech.: Res. Dev.* 2(1):23-42, January 1983.

[FREI84] Frei, H.P. and J. F. Jauslin. "Two-Dimensional Representation of Information Retrieval Services." In Dietschmann, H. J. (Ed.). *Representation and Exchange of Knowledge as a Basis of Information Processes.* New York: North-Holland, 1984, pp. 383-396.

[GENE85] Genesereth, M. R. and M. L. Ginsberg. "Logic Programming." *Communications of the ACM* 28(9):933-941, September 1985.

[GEOB88] Geobase, compilation of geographical facts, bundled with *Turbo Prolog 2.0* as demonstration Prolog file. Scotts Valley, CA: Borland International, 1988.

[GUID83] Guida, G. and C. Tasso. "An Expert Intermediary System for Interactive Document Retrieval." *Automation* 19(6):759-766, 1983.

[HAFN81] Hafner, C. D. "Representation of Knowledge in a Legal Information Retrieval System." In Oddy, R. N. et. al. (Eds.). *Information Retrieval Research.* London: Butterworths, 1981, pp. 139-153.

[HALL80] Hall, P. A. V. and G. R. Dowling. "Approximate String Matching." *ACM Comp. Surveys* 12(4):381-402, 1980.

[HAHN84] Hahn, U. and U. Reiner. "Heuristic Text Parsing in 'Topic': Methodological Issues in a Knowledge-Based Text Condensation System." In Dietschmann, H. J., (Ed.) *Representation and Exchange of Knowledge as a Basis of Information Processes.* New York: North-Holland, 1984, pp. 143-163.

[HARR85] Harris, M. D. *Introduction to Natural Language Processing.* Reston VA: Reston Publishing Co., 1985.

[HART86] Harter, S. P. *Online Information Retrieval: Concepts, Principles and Techniques.* Orlando FL: Academic Press, 1986.

[HASK80] Haskin, R. "Hardware for Searching Very Large Text Databases." In *Proc. 5th Workshop on Comp. Arch. for Non-Numeric Processing, New York: ACM, March 1980,* pp. 49-56.

[HAYE85a] Hayes-Roth, B. "A Blackboard Architecture for Control." *Artificial Intelligence* 26 (1985), pp. 251-321.

[HAYE85b] Hayes-Roth, F. "Rule-Based Systems." *Communications of the ACM* 28(9):921-932, September 1985.

[HELM85] Helm, A. R., Marriott, K. and C. Lassez. "Prolog for Expert Systems: An Evaluation." *Expert Systems in Government Symposium, Maclean, VA: Oct. 24-25, 1985.* IEEE, 1985, pp. 284-293.

[HORA84] Horak, W. and G. Kronert. "An Object-Oriented Office Document Architecture Model for Processing and Interchange of Documents." In *Proceedings of the Second ACM-SIGOA Conference on Office Information Systems, June 25-27, 1984,* pp. 152-160.

[HORA85] Horak, W. "Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization." *Computer* 18(10):50-60, October 1985.

[JANE89] *Jane's Fighting Ships.* London: Jane's Publishing Co., Ltd., 1989.

[KACP89] Kacprzyk, J., Zadrozny, S. and A. Ziolkowski. "FQUERY III+: A 'Human-Consistent' Database Querying System Based on Fuzzy Logic with Linguistic Quantifiers." *Information Systems* 14(6): 443-453, 1989.

[KHAN88] Khan, M. R. "Support Routines for the CODER System." Masters Project. Blacksburg VA: Virginia Tech Department of Computer Science, 1988.

[KIMW90] Kim, W., Garza, J. F., Boole, N. and D. Woelk. "Architecture of the ORION Next-Generation Database System." *IEEE Transactions on Knowledge and Data Engineering* 2:109-124, March, 1990.

[LANC78] Lancaster, W. F. *Toward Paperless Information Systems.* New York: Academic Press, 1978.

[LEBL89] LeBlond, G. T., LeBlond, W. B., Heslop, B., Polenberry, S. and C. Post. *dBASE IV: The Complete Reference.* Berkeley CA: Osborne McGraw-Hill, 1989.

[LEHN82] Wendy G. Lehnert and Martin H. Ringle, (Eds.) *Strategies for Natural Language Processing.* Hillsdale NJ: Lawrence Erlbaum Assoc., 1982.

[LENA86] Lenat, D., Prakash, M. and M. Shepherd. "CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks." *AI Magazine* 7(4): 65-84, Winter 1986.

[LENA90] Lenat, D., Ramanathan, V., Pittman, K., Pratt, D. and M. Shepherd. "CYC: Toward Programs with Common Sense." *Communications of the ACM* 33(8):30-49, August 1990.

[LEEN85] Lee, N. S. "P-Shell: A Prolog-Based Knowledge Programming Environment." Holmdel, NJ: AT&T Technical Report No. TM 59567-851201-01, December 1985.

[LOAT88] Loatman, R. B. and Post, S. D. "A Natural Language Processing System for Intelligence Message Analysis." *Signal: The International Journal of C^3I,* Sept. 1988.

[LUME89] *LUMENA™ 16/32 Version 3.3 User's Manual.* Santa Rosa, CA: Time Arts, Inc. 1989.

[LUTH90] Luther, A. C. *MEDIAscript™ Multimedia Application Design Tools: User Manual.* Springfield, VA: Network Technology Corporation, 1990.

[MACL90] Macleod, I. A. "Storage and Retrieval of Structured Documents." *Information Processing & Management* 26(2): 197-208, 1990.

[MALO87] Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A. and M. D. Cohen. "Intelligent Information-Sharing Systems." *Communications of the ACM* 30(5):390-402, May 1987.

[MARC88] March, S. T., (Ed.) *Entity-Relationship Approach: Proceedings of the Sixth International Conference on Entity-Relationship Approach, New York, USA, November 9-11, 1987.*

[MARC89a] Marchionini, G. "Information-Seeking Strategies of Novices Using a Full-Text Electronic Encyclopedia" *JASIS* 40(1): 54-66, 1989.

[MARC89b] Marcaccio, K. Y. *Computer-Readable Databases: A Directory and Sourcebook, Fifth Edition.* Detroit MI: Gale Research, Inc., 1989.

[MEAD81] Meadow, C. T. and P. A. Cochrane. *Basics of Online Searching.* New York: John Wiley & Sons, 1981.

[MEAN88] Means, L. G. "CN YUR CMPUTR RAED THS?" *Proceedings of the Second Conference on Applied Natural Language Processing, (Austin, TX: Feb. 9-12, 1988),* Association for Computational Linguistics, *CACL, 1988, pp. 93-100.*

[MINS75] Minsky, M. "A Framework for Representing Knowledge." In Winston, P. (Ed.). *The Psychology of Computer Vision.* New York: McGraw-Hill, 1975.

[MINS81] Minsky, M. "A Framework for Representing Knowledge." In J. Haugeland (Ed.). *Mind Design.* Cambridge MA: The MIT Press, 1981, pp. 95-128.

[NAIS83] Naish, L. "The MU-Prolog Deductive Database." Technical Report 83/18, Melbourne, Australia: University of Melbourne Department of Computer Science, November 1983.

[NAVA89] Navathe, S. B. and R. Ahmed. "A Temporal Relational Model and a Query Language." *Information Sciences* 49: 147-175, Oct. 1989.

[NIIP86a] Nii, P. H. "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures." *AI Magazine* 7(2):38-53, Summer 1986.

[NIIP86b] Nii, P. H. "Blackboard Systems: Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective." *AI Magazine* 7(3):82-106, August 1986.

[OCON80] O'Connor, J. "Answer-Passage Retrieval by Text Searching." *J. Am. Soc. Inf. Sci.* 31(4):227-239, 1980.

[ODDY77] Oddy, R. N. "Information Retrieval Through Man-Machine Dialogue." *J. Doc.* 33(1):1-14, March 1977.

[PAIC84] Paice, C. D. "Soft Evaluation of Boolean Search Queries in Information Retrieval." *Inf. Tech.: Res. Dev. Applications* 3(1):33-42, 1984.

[PATE84a] Patel-Schneider, P. F., Brachman, R. J. and H. J. Levesque. "ARGON: Knowledge Retrieval Meets Information Retrieval." *The First Conference on Artificial Intelligence Applications, Denver, CO: Dec. 5-7, 1984,* IEEE, 1984, pp. 280-286. (Also: Fairchild Technical Report No. 654; FLAIR Technical Report No. 29, September 1984).

[PATE84b] Patel-Schneider, P. F., "Small can be Beautiful in Knowledge Representation." *Workshop on Principles of Knowledge-Based Systems, Denver, CO: Dec. 3-4, 1984,* IEEE, 1984, pp. 11-16. (Also: FLAIR Technical Report No. 37, October 1984).

[PEEL85] Peels, A. J. H. M., Janssen, N. J. M. and W. Nawijn. "Document Architecture and Text Formatting." *ACM Transactions on Office Information Systems* 3(4):347-369, October 1985.

[POLL87] Pollitt, S. "CANSEARCH: An Expert Systems Approach to Document Retrieval." *Information Processing & Management* 23(2):119-138, 1987.

[QUIN91a] *Quintus Prolog I: Programming Environment.* Palo Alto CA: Quintus Corp., 1991.

[QUIN91b] *Quintus Prolog II: Language and Library.* Palo Alto CA: Quintus Corp., 1991.

[QUIN91c] *Quintus Prolog III: Interfaces and Delivery Tools.* Palo Alto CA: Quintus Corp., 1991.

[QUIN91d] *Quintus Prolog IV: Reference Pages.* Palo Alto CA: Quintus Corp., 1991.

[QUIN91e] *Quintus X Window Interface.* Palo Alto CA: Quintus Corp., 1991.

[QUIN91f] *Quintus Prolog Release 3.0 (DECstation).* Palo Alto CA: Quintus Corp., 1991.

[RAUL89] Rau, L. R., Jacobs, P. S. and U. Zernik. "Information Extraction and Text Summarization Using Linguistic Knowledge Acquisition." *Information Processing and Management* 23(4): 419-428, 1989.

[RIEG81] Rieger, C. and S. Small. "Toward a Theory of Distributed Word Expert Natural Language Parsing." *IEEE Trans. on Systems, Man and Cybernetics,* SMC-11(1):43-51, Jan. 1981.

[RIDJ88] Ridjanovic, D. Jarvenpaa, S., Mantha, R., Ram, S. and A. Rosenthal. "ER versus Relational: What Are the Differences?" in March, S. T., (Ed.) *Entity-Relationship Approach: Proceedings of the Sixth International Conference on Entity-Relationship Approach. New York, USA, November 9-11, 1987.*

[RITC74] Ritchie, D. M. and K. Thompson. "The UNIX Time-Sharing System." *Communications of the ACM* 17:(7):365-375, July 1974.

[ROBE76] Robertson, S. E. and K. S. Jones. "Relevance Weighting of Search Terms." *J. Am. Soc. Inf. Sci.* 27(3):129-146, 1976.

[ROCC71] Rocchio, Jr., J. J. "Relevance Feedback in Information Retrieval." In G. Salton (Ed.), *The SMART Retrieval System, Experiments in Automatic Document Processing.* Englewood Cliffs NJ: Prentice-Hall, 1971, chapter 14.

[ROWE88] Rowe, N. C. *Artificial Intelligence through Prolog.* Englewood Cliffs NJ: Prentice-Hall, 1988.

[SACC84] Sacco, G. M. "OTTER - An Information Retrieval System for Office Automation." In *Proc. Second ACM-SIGOA Conf. on Office Information Systems, June 25-27, 1984,* pp. 104-112.

[SALT83a] Salton, G. and M. J. McGill. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill, 1983.

[SALT83b] Salton, G., Buckley, C. and E. A. Fox. "Automatic Query Formulations in Information Retrieval." *J. Am. Soc. Inf. Sci.* 34(4):262-280, July 1983.

[SALT83c] Salton, G., Fox, E. A. and H. Wu, "Extended Boolean Information Retrieval." *Communications of the ACM* 26(11):1022-1036, Nov. 1983.

[SALT85] Salton. G., Fox, E. A. and E. Voorhees. "Advanced Feedback Methods in Information Retrieval." *J. Am. Soc. Inf. Sci.* 36(3):200-210, May 1985.

[SANC89] Sanchez, E. "Importance in Knowledge Systems." *Information Systems* 14(6): 455-464, 1989.

[SCHA81] Schank, R. C., Kolodner, J. L. and G. DeJong. "Conceptual Information Retrieval." In R. N. Oddy et al. (Eds.), *Information Retrieval Research.* London: Butterworths, 1981.

[SCHE82] Scheuermann, P. *Improving Database Usability and Responsiveness.* New York: Academic Press, 1982.

[SCHN78] Schneiderman, B., (Ed.) *Databases: Improving Usability and Responsiveness.* New York: Academic Press, 1978.

[SHAP79] Shapiro, S. C. "The SNePS Semantic Network Processing System." In Findler, N., (Ed.) *Associative Nextworks.* New York: Academic Press, 1979, pp. 179-203.

[SIMM84] Simmons, Robert F. *Computations from the English.* Englewood Cliffs NJ: Prentice-Hall, 1984.

[SKUC85] Skuce, D., Matwin, S., Tauzovich, B., Szpakowicz, S. and F. Oppacher. "A Rule-Oriented Methodology for Constructing a Knowledge Base from Natural Language Documents." *Expert Systems in Government Symposium. Maclean VA: Oct. 24-25, 1985.* IEEE, 1985, pp. 378-385.

[SMIT84] Smith, L. C. and A. J. Warner. "A Taxonomy of Representations in Information Retrieval System Design." In Dietschmann, H. J. (Ed.). *Representation and Exchange of Knowledge as a Basis of Information Processes.* New York: North-Holland, 1984, pp. 31-49.

[SMIT87] Smith, L. C. "Artificial Intelligence and Information Retrieval." *Annual Review of Information Science and Technology* 22:41-77, 1987.

[SPAR84] Sparck-Jones, K. and J. I. Tait. "Automatic Search Term Variant Generation." *Journal of Documentation* 40(1):50-66, March 1984.

[SPAR87] Sparck-Jones, K. "Architecture Problems in the Construction of Expert Systems for Document Retrieval." Paper presented at the Workshop on Distributed Expert-Based Information Systems. School of Communication, Information and Library Studies, Rutgers University, March 1987.

[SRID87] Sridharan, N. S. "Report on the 1986 Workshop on Distributed Artificial Intelligence." *AI Magazine* 8(3):75-85, Fall 1987.

[STEE90] Steele, P. and R. Heydt. *Professional dBASE IV 1.1 and SQL Programming.* New York: John Wiley & Sons, 1990.

[STER86] Sterling, L. and E. Shapiro. *The Art of Prolog.* Cambridge MA: The MIT Press, 1986.

[SUBR85] Subrahmanyam, P. A. "The 'Software Engineering' of Expert Systems: Is Prolog Appropriate?" *IEEE Trans. on Software Engineering* 11(11):1391-1400, November 1985.

[TENO84] Tenopir, C. "Full-Text Databases." *ARIST* 19:215-246, 1984.

[THOM85] Thompson, R. H. and W. B. Croft. "An Expert System for Document Retrieval." *Expert Systems in Government Symposium. Maclean VA: Oct. 24-25, 1985.* IEEE, 1985, pp. 448-456.

[THOM87a] Thom, J. A. and J. Zobel, (Eds.) "Extracts from NU-Prolog Reference Manual," Version 1.1. Technical Report 86/10. Victoria, Australia: University of Melbourne, Department of Computer Science, May 1987.

[THOM87b] Thom, J. A. and J. Zobel, (Eds.) "NU-Prolog Reference Manual." Victoria, Australia: University of Melbourne, Department of Computer Science, May 1987.

[TONG83] Tong, R. M. et al. "A Rule-Based Approach to Information Retrieval: Some Results and Comments." *Proceedings of the National Conference on Artificial Intelligence (Washington, DC, August, 1983) AAAI,* pp. 411-415, 1983.

[TONG85] Tong, R. M., Askman, V. N., Cunningham, J. F. and C. J. Tollander. "RUBRIC, An Environment for Full Text Information Retrieval." *Proc. 8th Annual Int. ACM Conference on R&D in Inf. Ret., Montreal, June 1985.*

[WEAV88a] Weaver, M. T. and E. A. Fox. "Implementing an Intelligent Retrieval System: The CODER System, Version 1.0." MS Thesis. Blacksburg VA: VPI&SU Computer Science Dept., February 1988. (Available from VPI&SU as TR-88-6.)

[WEAV88b] Weaver, M. T., R. K. France, Q-F. Chen and E. A. Fox. "A Frame-Based Language in Information Retrieval." TR-88-25, Virginia Tech Dept. of Comp. Sci., 1988.

[WHIT83] White, C. "The Linguistic String Project Dictionary for Automatic Text Analysis." In *Proc. Workshop on Machine Readable Dictionaries, SRI, Menlo Park CA, May 1983.*

[WINE85] Winett, S. and E. A. Fox. "Using Information Retrieval Techniques in an Expert System." *The Second Conference on Artificial Intelligence Applications Miami Beach FL: Dec. 11-13, 1985.* IEEE, 1985, pp. 230-235.

[WOHL86] Wohlend, R. C. "Creation of a Prolog Fact Base from the HAI." Master Project Report. Blacksburg VA: Virginia Tech Department of Computer Science, March 1986.

VITA

Richard Dee Barnhart

Born: Feb. 3, 1944

Education:

B. S., Whitworth College, 1966, major in Mathematics

M.S., University of Idaho, 1968, major in Mathematics
Thesis: "Continuous Curves"

Ph. D., University of Idaho, 1972, major in Mathematics
Dissertation: "A Class of 2-complexes Which Are Spines of a 3-ball"

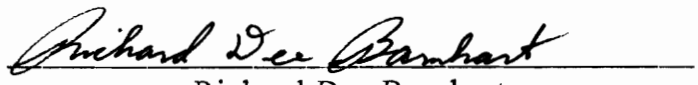
Employment:

Bryan College, Associate Professor of Mathematics, 1971-78

KMC Grocery Company, Programmer 1978-80,
Data Processing Manager, 1980-82.

Management Software, Programmer and Consultant, 1982-83.

Liberty University, Associate Professor of Mathematics and Computer Science, 1984-1992,
Professor of Computer Science, 1992-.



Richard Dee Barnhart