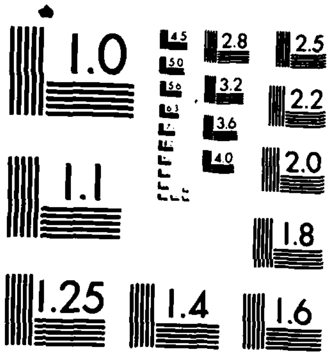
AD-A162 181 THE DESIGN OF NATURAL LANGUAGE INTERFACES(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA I KOTROZOS SEP 85 1/2

UNCLASSIFIED F/G 5/8 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

(2)

AD-A162 181

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
DEC 13 1985
S D
D

THESIS

THE DESIGN OF NATURAL LANGUAGE INTERFACES

by

Ioannis Kotrozos

September 1985

Thesis Advisor: Tung X. Bui

Approved for public release; distribution is unlimited

85 12 12 054

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. ADA 162 181	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Design of Natual Language Interfaces		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1985
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ioannis Kotrozos		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100		12. REPORT DATE September 1985
		13. NUMBER OF PAGES 109
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited

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

18. SUPPLEMENTARY NOTES

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

man-machine interfaces; natural languages; man-machine communication.

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

One of the possible solutions to the problem of designing effective man-machine interfaces seems to be the use of natural languages.
This thesis examines the principles of design of effective man-machine interfaces, the role of natural languages in achieving effective man-machine communication and the implementation issues and techniques for their use as interfaces.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The Design of Natural Language Interfaces

by

Ioannis Kotrozos
Lieutenant, Hellenic Navy

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

Author: [illegible]
Ioannis Kotrozos

Approved by: Tung Bui
T.X. Bui, Thesis Advisor

Barry Frew
B.A. Frew, Second Reader

[illegible]
W.R. Greer Jr, Chairman,
Department of Administrative Sciences

K.T. [illegible]
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

One of the possible solutions to the problem of designing effective man-machine interfaces seems to be the use of natural languages.

This thesis examines the principles of design of effective man-machine interfaces, the role of natural languages in achieving effective man-machine communication and the implementation issues and techniques for their use as interfaces.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. BACKGROUND

Until recently it was required that the computer user should be a specialist with an appropriate education for handling the computer without causing serious problems. The requirements of the contemporary enterprise for as many users as possible and the advance of the technology have brought many people in front of the computer not only for professional but also for private use. The problem today is how these people would be able to take as much information as possible out of the data available in the computer's storage devices.

Two major aspects have to be considered in the design of man-machine communication; the characteristics of the user and the means of interaction between him and the computer. The study of user characteristics helps the designer establish objectives for the man-machine dialogue and the principles which govern the dialogue at the interface. Not all means of interaction are appropriate for all cases. For example, programming languages are inappropriate for users who have not learned programming and who probably would not be eager to do this, for example, administration staff. Natural languages are inappropriate where precise input interpretation is required.

The cost factor is the most serious factor for the selection of an interface which takes into account the user and his needs. Another important factor is the variety in the features of the users which may result in the fact that a specific type of dialogue is appropriate for some of them but it is not appropriate for others.

A man-machine interface is the medium of interaction between man and machine. For example as the steering wheel of a car is an interface where the driver and the car interact, the terminal is an interface between the user and the computer . The interface may also be something abstract, like a language, which is the medium for transferring user ideas to the computer, which are then translated by the computer into appropriate actions.

Given that a user's statement may have more than one meaning with respect to various sets of actions, it is possible that each set of actions will require its own interface.

The requirements of the contemporary enterprise for users who are not specialists or experts in interacting with the computer, ask for interfaces which are user-friendly and are addressed to the average user. The design of these interfaces must take into account the needs of the average user as a human being and the user's characteristics and behavior when he faces the computer.

This thesis concerns the category of the so called "linguistic" interfaces which includes the languages which are used as interfaces between man and computer. Natural Language seems at first look to meet the objectives for the design and implementation of a "humanized" interface which addresses the needs of the average user.

A large number of natural language interfaces have been developed today. Yet, many practical questions remain unsolved. There are many arguments about the best architecture of natural language interfaces or whether these interfaces are preferable to the artificial language interfaces designed with human factors in mind. Besides the problem for evaluating the best alternative there are some other questions which ask for an answer:

- Is natural language implementation feasible ?
- If natural language interfaces can successfully be implemented, do they solve the problem ?

- How transportable are they to new applications ?

The purpose of the thesis is to search the design issues and implementation techniques of Natural Languages as man-machine interfaces and to present a possible design of an acceptable Natural Language Interface.

In the first chapter of the thesis a comparison of natural languages to the existing programming and data retrieval language is made. The next two chapters examine the objectives and design principles of man-machine interfaces in general and provide suggestions about the requirements which man-machine interfaces must satisfy. The next chapters concern the implementation issues and techniques for the natural languages, their practicality and suggest a design for a general purpose portable Natural Language Interface. In the last chapter, the possibilities for promoting natural language applications in the military environment are examined.

II. NATURAL LANGUAGES VS COMPUTER LANGUAGES

A computer performs its various functions by means of a program of instructions which is the target program. The various languages are the means by which people can communicate their ideas to the target program to make this program do what they desire. A person must generate a statement in a certain language. This statement is then communicated with the program, which in turn will translate the statement to appropriate actions.

Hundreds of programming languages are available today. These languages are categorized in levels, with respect to how closely they represent the instructions in the form they take when they are in the computer central processor, which is called in the literature "machine code". In the lower level, there are those languages which are similar to the machine code and which are machine dependent. In the next higher level are the so called high level languages, which are problem oriented rather than machine oriented as lower level languages are. The higher level languages were developed to solve the problem of program compatibility between the different types of machines. Programs written in high-level language can be in some degree read and understood by someone who has not learned the language. Commercial languages resemble highly artificial but still recognizable, English; scientific languages resemble mathematical notation. These high level languages still require that a compiler be written for each machine on which the program is to be run. The fact that it is very common that several features of a specific language will be modified for several reasons, makes it possible for several "dialects" of the same language to exist. The attempts are still directed towards the production of a language which is sufficiently general for all purposes, can be understood by almost anybody and is sufficiently simple for the production of compilers of identical versions on all types of machines. The problem of ideal communication has still not been resolved although enough progress has been made.

A. CHARACTERISTICS OF PROGRAMMING LANGUAGES

The software psychologist Gerald Weinberg [Ref. 1], determined a number of psychological characteristics which exist in the programming languages, in various degrees.

1. Uniformity

Uniformity characterizes the degree to which a language uses consistent notation, applies arbitrary restrictions and supports syntactic or semantic exceptions to the rule. For example ADA's "overloading" permits the use of identifiers and operators which may have alternative meanings within the same scope.

2. Ambiguity

Ambiguity characterizes the degree to which a statement is interpreted in different ways by different programmers although the compiler will always interpret that statement in one way. For example the arithmetic statement

```
Z = A + B / C * D
```

may be interpreted by one user as Z = A + (B / C) * D and by another as Z = A + B / (C * D).

3. Compactness

Compactness of a programming language characterizes the amount of coded information that must be recalled from the human memory. Some compactness measures are the variety of data types, operators, built in functions etc. APL for example which permits little code to be used for arithmetic and logical operations, exibits a high degree of compactness.

B. HUMAN MEMORY AND LANGUAGE CHARACTERISTICS

Human memory characteristics strongly affect the way we use a particular language. Human memory is divided into synesthetic and sequential domains [Ref. 2:pp. 270]. The first, allows us to remember and recognize things as a whole; the second, helps to recall the next thing in a sequence. Both these characteristics have a strong effect to two other language characteristics the locality and linearity. Locality is related to the degree to which the statements can be combined to block, support the structured constructs and code modularity and cohesiveness. Linearity is accociated with the degree that a sequence of logical operations facilitates human perception.

The ability of a human to learn a new language is determined by the characteristic of tradition. For example a person with an ALGOL background should not be expected to have any strong difficulties in learning PASCAL. The same person will probably have problems in learning PROLOG or LISP.

C. CHARACTERISTICS OF NATURAL LANGUAGES

Almost all programming today, is done in high-level languages which to some degree are designed with the human in mind. However, high-level languages always remain highly stylized with strict rules which have to be learned and followed. This formalized type of dialogue is in many cases undesirable. A computer programmer often feels the need to give instructions to the computer as if he had given the same instructions in another person.

Considering Natural Language characteristics and artificial languages characteristics both similarities and big differences can be found.

1. Humanized Syntax

The presence of human syntax is the most significant characteristic of Natural Language. There are numerous words and word combinations which are used by human beings and found in a Natural Language.

2. Disordered Word Arrangement

Another important characteristic is that words in Natural Language are used disorderly and they are very often used for different purposes. They may for example be at one time verbs and the other time, nouns.

3. Use of Imperfect Expressions

Imperfect expressions of thought may be used and are "legal" for the creation of sentences.

4. Presuposition

Another characteristic is that one of presuposition [Ref. 3:pp. 40]. This characteristic is a result of the thought that the recipient of the instruction has mental processes which are close to those of the person who generates the instruction.

5. Flexibility and Ease of Learning

These are the two most attractive features of Natural Language because they provide the user with freedom in constructing his statements and minimize the mental load for learning them.

D. AMBIGUITY IN NATURAL LANGUAGES

Many words and sentences of Natural Language are ambiguous. This means that they may have more than one meaning. Even if an entire statement is unambiguous its parts may be ambiguous with respect to the context. To make the computer recognize the meaning of Natural Language statements requires programs of very high complexity and this is rather impractical in the majority of today's serial machines.

1. Double Negatives

Another case of ambiguity occurs when double negatives are encountered in the statement. A characteristic example is the problem which arises with the rules of Boolean Algebra. In conventional programming languages the expression "NOT FALSE" can be replaced by "TRUE" and the expression "NOT TRUE" by "FALSE". This cannot be done in Natural Language.

2. Similarity of Wording

In many cases similar expressions may have different meanings. For example the expressions

A book of J. G. Norton

and

A book of Physics

have similar wording but different meaning; The first means "a book written by J. G. Norton" and the second means "a book whose subject is physics".

3. Time of Statement Execution within the Program

Ambiguity also occurs when no distinction is made to various instructions with respect to the time of execution. This does not happen in programming languages where there is clear distinction between these instructions which have to be executed in the main program and those to be executed in subroutines. In addition, while in programming languages conditional instructions and conditions are clearly stated this is not always done in natural language. For example, in the following series of activities

```
study chapter #5
do exercises in chapter #4
go for a walk
```

no clear distinction is made about the desired time of execution of each statement with respect to the others.

4. Use of "AND", "OR", "NOT"

The use of "AND", "OR" and "NOT" is another case where confusion may be created because sometimes the "OR" may be conceived as inclusive and other times as exclusive, depending on the context. This of course does not happen in programming languages where their use is precisely defined.

5. Pronouns

Pronouns are also a potential source of ambiguity. Their presence generates serious interpretation problems because determining what they refer to can be defined only with the help of the context.

6. Arithmetical Expressions

Arithmetic expressions are another source of ambiguity. Interpretation of numbers and arithmetical operations in natural language is not always a simple task. For example when we say

France produces 2000 Mirage airplanes

we do not specify if "2000" defines the model number or the quantity produced.

7. The "dangling" ELSE

A common ambiguity problem to both programming and natural languages is the problem of "dangling else" where we cannot specify to which "if" the "else" refers to. The difference is that although various programming languages provide various solutions to this problem, natural languages do not. For example, if we consider the series of program instructions displayed on Figure 2.1, we cannot specify to which "IF", "ELSE" refers to.

```
If x > 0 then      ? <-----
y = sqrt(x)                |
If x < 0 then      ? <---- |
y = i * sqrt(-x)           |
Else    >---------------
y = 0
```

Figure 2.1 Dangling "ELSE".

E. NATURAL AND NON-PROCEDURAL PROGRAMMING LANGUAGES

There are some cases of natural language dialogue where possible misinterpretation is unimportant. If the conditions require a precise natural language input, the user must be confined in what he wants to say to a small vocabulary whose words are precisely defined. This means that there is always a trade-off between dialogue flexibility and preciseness. Humans are able to establish a meaning even for incomplete statements by examining those statements in context of their own view of the world. It is very difficult to provide the computer with such a huge context where all possible statements can be referenced.

In most of the practical applications today, the vocabulary used is limited. In these cases, natural language tends to be structured by employing formal structures which are precisely defined. After this, natural language is not as much effective or even acceptable, as a programming language. There is today a trend to transit from procedural languages like COBOL, PL1 to non-procedural languages like FOCUS, INFO etc. A common feature of non-procedural programming languages and programming languages is that no precise step-by-step instructions are required but in some way a specification of the desired result. The functionality of the non-procedural languages is so high that most organizations tend to adopt them.

F. NATURAL LANGUAGE IN DATABASE QUERYING

Natural language may also be used to retrieve information from computerized databases. The use of natural language in database query is directly related to the benefit of taking advantage of the database processing (more information out of a given amount of data, increased user satisfaction etc.).

Two solutions exist basically to the problem of database querying:

- Adapt the user to the machine by having him learn a formal artificial querying language.
- Adapt the machine to the user by making it able to understand a section of natural language.

The first solution presents several disadvantages which cannot be ignored. The most significant is the cost involved in training users to learn formal querying languages. Beyond this problem there is another significant issue. The fact that a formal language will be used does not eliminate the dangers of ambiguity where various users will state their requirements in their own different ways.

The value of a natural language query is directly related to how effectively natural language can be dealt with a particular application. Natural language contributes to the understandability of the database queries by:

- Providing capability of selecting subsets of file records with respect to the desired criteria.
- Reflecting the data manipulation process.
- Reflecting user's logical view of data.

Measuring user's satisfaction with a specific natural language querying system is today possible and several commercial systems have been tested so far. The results of these tests show that some elementary and short English forms have been found by the users to be more convenient than natural language queries. A recent experiment [Ref. 3] for the evaluation of the desirability of natural language queries as compared to a formal query language like SQL concluded that:

- SQL is for the moment superior to natural language.
- Natural language is briefer than SQL.
- Natural language does not provide user with adequate feedback.

III. THE DESIGN OF EFFECTIVE MAN-MACHINE INTERFACES

A. DESIGN STEPS

For the design of the man-machine interface several ideas may be considered and there are several steps that the designer should go through [Ref. 4:pp. 9]. Before proceeding further, it is useful to look at the design steps of the man-machine dialogue to provide a context for the discussion of various user and dialogue aspects:

- Organization of information flow: To select a configuration for information exchange between the user and the computer. To see for example, if the user will communicate with the machine directly or through an experienced operator.
- Selection of a general dialogue category: To determine a category. To see for example, if a programming language is going to be used instead of a limited English input (see Table I).
- Hardware requirements specification: To relate dialogue requirements to system configuration.
- Dialogue structuring: To determine various formats and other details.
- Establishment of error procedures: To establish procedures for minimizing the likelihood of errors and when errors occur to maximize the possibility of detecting them. To provide also means for minimizing the consequences of machine errors.
- Dialogue simulation: To assure that the dialogue will work well with the specific users.
- Dialogue immunization: To make programs invulnerable to irrational user behavior.

TABLE I

Dialogue Categories

i. Artificial Programming Languages

Advantages	:	Precise, concise
Disadvantages	:	Inappropriate for users with low programming experience.

ii. Natural Languages

Advantages	:	Easy to learn and use
Disadvantages	:	Unsuitable where precision is required.

iii. Limited Natural Language Input

Advantages	:	Familiarity with the vocabulary.
Disadvantages	:	Overestimation of the real understanding capabilities of the system.

iv. Computer-initiated Dialogue

Advantages	:	Little user training, very low error possibility.
Disadvantages	:	Dialogue is long and very slow. User is restricted to a finite number of predefined options.

V. User-initiated Dialogue

Advantages	:	Simple for the user.
Disadvantages	:	Low reliability, limited scope and flexibility

vi. Light-pen Input

Advantages	:	Simple, good for inexperienced users.
Disadvantages	:	Limited scope.

vii. Graphics with Charts Displays

Advantages	:	Effective representation of the substance of the information.
Disadvantages	:	Expensive.

B. THE NEEDS OF THE USER

It is very important in the initial steps of the dialogue design, that user needs and characteristics will be identified. These will provide a frame on which to taylor the dialogue. It is possible that more than one dialogue type and special feature of each type will be found to be appropriate and that the designer should consider all possibilities before deciding upon a particular type and structure.

By examining various user types which are frequently mentioned in the literature [Ref. 5], five types are usually refered to: the casual users, parametric users, analysts, application programmers and systems programmers. These types are classified according to (i) the ability to handle the system, and (ii), by how often and how long they make use of it. It is important to notice here that user types should be seen as a set of information needs rather than as a set of persons.

Another area we should look at is the user's behavior at the interface. In doing this we consider the factors which affect this behavior from two viewpoints:

- The user as a human being who has certain needs.
- The user facing the system.

Beginning with the first, user has needs which must be satisfied. Under this assumption the user probably wants to be in front of a user-friendly system which does not force him to act in a strictly predetermined way and does not restrict him in his communication.

User's behavior in front of the system is affected by his data processing and application background, his attitude towards the system, his objectives and the task structure.

C. USER ORIENTED OBJECTIVES OF A MAN-MACHINE INTERFACE

The primary concern is always to design an interface whose goal is to adapt the computer to the user and not to force the user to adapt to the computer.

Considering the user as a human being the objectives shall be directed toward his or her general needs, skills and characteristics:

- Systems behavior towards user must be flexible so that user will be provided with a certain degree of freedom in his actions.
- The system must be adaptable to the various user types and interaction modes.
- User must not be concerned too much with system's behavior.
- The system must never cause frustration to the user.
- The man-computer interaction must be as similar as possible to the communication between humans.
- The system must be able to interact with the average user who does not have special abilities.
- System's behavior must not be contradictory, so the user is allowed to easily get accustomed with system's operation.
- The dialogue must be performed in a way which enables user to gradually pass into a more sophisticated type of dialogue.
- The system must be as insensitive as possible to errors made by the user and must not introduce errors by itself.

Considering the user facing the system, the objectives shall address the factors which affect his behavior:

- System's use must be possible for users who do not have special education.

- The system must be able to interact with users having different levels of experience.
- Different systems must have interfaces which are standardized.
- System must not cause disappointment.
- System's handling must not turn user's attention from the actual problem that he uses the computer used for .
- System's use must not be restricted to specific cases but it must allow the user to modify the communication according to the requirements.

D. TECHNICAL ISSUES OF MAN-MACHINE DIALOGUE DESIGN

The term man-computer dialogue means the interactive exchange of messages between the user and a dialogue system, according to an agreed dialogue language and a dialogue category, to accomplish a desired task. A basic principle of the man-computer dialogue is that the user has no restrictions on the selection of input and system's reaction is conclusive. The means for the exchange of messages between the man and the computer are the dialogue modes. The five best known dialogue modes are :

- Commands.
- Menu.
- Form filling.
- Yes/No questions.
- Natural language interaction.

The properties of a basic dialogue model have been characterized [Ref. 5:pp. 47], as follows:

- In order for a dialogue model to be applicable, this model must be considered separately from the following details of :
  - Distribution of roles of dialog initiator and dialogue responder.

- Special subject matters and objectives of dialogues.
- Special input/output techniques and devices.

• For the different addresses of the dialogue descriptions the modes must be :
  - Representable in a simple and expressive way (user).
  - Accessible to formal treatment (theorist).
  - Appropriate for transformation into programs (implementor).

Petri-nets which were developed for representation of non-sequential processes, meet the requirements of the previous model satisfactory. They are considered very good for the representation of the different faces of the dialogue. Refinement and generalization of nets allow representation of certain dialogue aspects in the desired level of detail.

The more general level of the dialogue concerns the representation of the cooperation between information processing units and information communication units. Information processing units are the initiator and the responder/recipient of information and information communication units are the channels through which the information is transmitted. The next lower level is concerned with dialogue processes which are described in general. The last level, describes processes in detail and illustrates relationships between actions of dialogue participants and the messages which they communicate.

The basic unit of dialogue between two participants is the dialogue step which consists of processes and messages of the one participant followed by the processing and messages of the other. Petri-nets allow us to represent sequences of dialogue steps. The different dialogue types can be characterized from the role of action and reaction in the dialogue step. Combination of distinguishing features allow the outlining of the basic structure of all the possible dialogue steps. Description of the messages by which action and reaction are represented helps to distinguish among various dialogue types. A sequence of dialogue types specifies a particular dialogue method.

The human uses the computer for a certain purpose. He/she formulates tasks for the computer and the computer in turn corresponds to the assigned tasks which reflect the subgoals which the user sets for resolving a particular problem. The computer executes an indicated task by a set of activities which user cannot see. The user only knows the set of functions of the system which are realized by means of those activities. The system's functions are the abstraction of tasks and activities. When the system chooses a task the user is directed from the available functions. When he enters his task the computer identifies the required function and executes the corresponding activities.

There are six categories of functions [Ref. 5:pp. 60].

- Dialogue organization functions : help to organize the session at the terminal.
- Control functions : enable the user to control the processing sequence of his tasks.
- Input/output functions : apply to the user's terminal.
- Help functions : provide help in case of difficulties.
- Communication functions : enable the user to interact with other users of the system through the terminal.
- Extension functions : help the user to perform similar complex tasks which are repeated very often.

To support user's orientation during the dialogue, the system must be able to remember the past dialogue and know about the available functions and the actual dialogue type. In other words the system must know the dialogue state and the data which the dialogue handler has stored. The user actually is concerned only with the available functions. The system's design must allow the user to choose the dialogue mode and to get information about the dialogue past.

E. ERRORS IN MAN-MACHINE DIALOGUE

Error occurrence is a common case in man-machine communication. This happens even with the most experienced users. Errors are not usually considered as something special but they very often have very serious consequenses to both the system and the user. A great part of these errors come from the lack of sensitivity on the part of the system designer to the needs and abilities of the user.

Errors often belong in two broad categories. The first includes the errors concerning the erroneous implementation of a correct user intention. The other includes the errors concerning the implementation of an incorrect user intention. If we examine those errors with respect to their sources we will identify several classes which exist.
A large class of errors concerns the cases where the user believes that the system is in one state while it is actually in another. An example is the case where users give executive mode commands while they are in the edit mode. Another class of errors includes those errors where the user does the right operation in the wrong thing. They occur when ambiguity exists about which is the proper action and are caused when the action is not adequately specified. An example here, is the errors which occur in some text editors when some lower-case letters may cause the execution of different commands from those caused by the same upper-case letters. Some other errors are these which are caused when the user causes an action different from the intended because of insufficient knowledge about the proper operation he tries to derive the operation by analogy with something similar. The last class of errors are these where the

desired action cannot be carried out because of some hardware failure.

F. GENERAL RULES FOR INTERFACE DESIGN

This section presents a set of general rules based on what we have seen so far. Of course these rules are not absolute and in some cases they may overlap each other. The rules address two major goals of the interface design ; to minimize the mental load of the users and to provide the capability of early error detection, minimization of error consequences for both the system and the user and easy recovery:

- Mininimizing the mental load of the user:
  - All terminology and procedures should be uniformly available and consistently applied.
  - All information that the user needs should be easily and quickly accessed.
  - The information should reflect the actual use of the system.
  - Users should have some moderate "hands on" experience from some type of a model system before using their own.
  - Users should be given feedback about the state of the dialogue as often as possible.
- Minimizing the error correction and recovery effort:
  - System should provide means for validating data upon entry.
  - User should be allowed for backtrack through the dialogue sequence.
  - Different categories of actions should have have dissimilar command sequences.
  - The actions which may have serious consequences should be difficult to do.

- System should be tolerant of both its own and user errors.

In aa attempt to derive some possible general rules for the design of an effective interface, we examined the three major aspects for this task ; the user, the dialogue mechanism and the errors in the interaction between man and machine. Some general rules were at the end presented, based on the analysis of those three aspects.

These rules are by no means unique or absolute. Man-machine communication is a very sensitive and complex research area which consists of as the subject of interest not only the computer, but also and other sciences as cognitive psychology, ergonomics, etc. The fact that a human is involved as one component of the dialogue and the machine as the other makes it very difficult to create rules which are absolutely tailored to every type of communication.

IV. CONCEPTUAL RECOMMENDATIONS FOR MAN-MACHINE INTERFACES

A goal in the design of effective man-machine interfaces of the modern computer systems is that they will address the average user. This implies that they will be designed for users who are neither novices nor experts at computer handling. The goal of the design is to present an interface which takes into account the characteristics, skills and information needs of this type of user and his or her possible behavior in front of the machine.

Prior to examining the role of natural languages in the design of "humanized" interfaces we should look at the understanding, or in other words, process of communicating ideas between man and computer. When we say man-machine communication we mean the communication of ideas between a human and a program, which is in the computer in order to accomplish a specific purpose. In this process the ideas in the human mind are generators of statements which in turn communicate to a program which translates them into appropriate actions. In this context, a statement is said to be understood with respect to a given language and a target set of actions. In order for a statement to be understandable it must be associated with a single set of actions.

This is the point where the problems with the use of the natural language start. These problems come from the fact that more than one set of actions may be related to each statement. This implies more than one meaning for the same statement and this is the source of ambiguity. This requires that there be one-to-one relationships between programs and target sets of actions and in many cases these sets may be so broad that it is impossible to consider the entire user's statement.

In order to decide if any language provides an appropriate interface, it is necessary that the degree to which this language addresses the characteristics and needs of the user be taken into account. The role of natural languages in the effectiveness and efficiency of man-machine interfaces is examined here with recpect to various user and other general attributes

Two principles for the Man-Machine Interface have been identified; first, the interface must not force the user to adapt to the computer and second, the interaction must be designed having in mind the average user who has not any special capabilities in handling the computer.

There are though, many other questions which the designer of a man-machine interface should answer; What are the capabilities of the user? What is his/her data processing background? How is he/she expected to behave in front of the system? How often will he/she use the terminal? How intelligent is he/she? How much must he/she be trained?

All these characteristics will help the designer to address the major issues in the interface design, like the following:

- Functions which must be provided at the interface.
- Dialogue categories that user can use.
- Levels of dialogue that user is going to use.

A. REALIZATION OF MAN-MACHINE INTERFACE OBJECTIVES

Several objectives have been defined for the man-machine interface. For further discussion on how the objectives will be realized it was found appropriate that the most representative among them had to be discussed:

1. Ease of Learning

User must be able to learn the dialogue by himself without special assistance. This objective addresses the novice user. Learning the interface asks for the satisfaction of the following requirements:

- The first thing that user needs to know is the functions of the interface.
- The user must be first introduced to the basic functions. Then according to his/her desire and experience he/she may pass in a more sophisticated level of functions.
- A good practice is that the user is not "flooded" with unnecessary information and he/she must be left the initiative to request additional information when he considers it necessary. The creation of passive users should be avoided.
- The output must be presented in an understandable and meaningful form where all subjects are easily recognizable.
- The number of rules which govern the dialogue and which the user must memorize should be as small as possible.

The most attractive characteristic of natural language is that it makes learning of interface very easy.

2. Ease of Use

The system must not create confusion or disappointment in the user and it must be easy to handle:

- The first thing that every user wants to know is information about the state of dialogue.
- A large number of help functions must be provided.
- The request for information by the user and the system's response must resemble the natural language as much as possible.

- There must be a fairly large number of functions which allow the user to dominate the system.

3. Flexibility

The system is adaptable to the user and task characteristics:

- The system provides a large number of functions.
- The system provides a variety of dialogue types.
- The user is able to switch from one dialogue type to another and create the desired level of sophistication.

An important notice at this point is the trade-off between flexibility and complexity.

The lack of strict rules in the use of the natural language makes the user feel comfortable in front of the machine. This benefit though, very often becomes an illusion if the language makes use of "legal" and "illegal" statements. This means that the user is indirectly forced to learn the boundaries of the language use. For example, a natural language database querying system may consider as legal the statement

> Give the names of all CO's of USS Enterprise
> since 1980

whereas the same system may consider the statement

> Give the names of all CO's of USS Enterprise
> the last 5 years

as illegal. The query in this example is actualy satisfied because there is at least one statement which is acceptable. The problem starts at the point where the user must learn which are the illegal statements. So natural language is desirable if the subset which has been chosen is broad enough and the users are not obliged to learn which statements are legal and which are not.

4. Transparency

System's behavior is predictable; the user can predict what information the system is going to provide him.

5. Reliability

The system is designed in a way which allows the user to conduct a dialogue in reliable way and this asks for the satisfaction of the following requirements:

- System's behavior must allow the user to know every possible effect of his/her input to the system.
- System's reaction to possible errors must not affect the continuation of the dialogue.

Although natural language is not governed by such strict rules as the artificial languages it in reality may push the user to make several kinds of errors. One characteristic category of such errors is that which includes the errors caused because of the similarity in the expression of various commands in natural language and this will make the user get confused about which is the appropriate command to use.

6. Precision Requirements

The problem of ambiguity is an inherent attribute of natural language statements. The reason is that these statements seldom stand for a single meaning. In cases where high precision is required ambiguity makes natural language interfaces inappropriate. The problem of ambiguity may somehow be resolved by combining natural language with some artificial language.

7. Feedback to the User

The lack of precision which causes so many problems in the understanding of the statement is the source of another problem; it is not easy for the system to provide feedback to the user about errors in dialogue or the exact reason why the system cannot understand the statement.

8. Picture Description

A very important form of man-machine dialogue is the communication through pictures. The media for this type of communication differ in their effectiveness and in the type of information they carry. An obvious weakness of natural language is the inability to describe shapes, positions and graphs. So the natural language cannot substitute for media like charts, tabulations and other screen representations. Programs which manipulate graphical objects may have natural language interfaces only if they are integrated with appropriate graphics based communication tools.

9. Interface Standardization

In implementing the natural language interface it is necessary that each program must be associated with a single target set of actions. This implies that a great number of these sets must exist. This does not favor the standardization of the interfaces and minimization of training requirements for the users.

10. User Experience

The use of natural language implies that the operation of the interface is independent of user's experience in a specific artificial language. The same does not happen when an artificial language has to be used. In this case user's experience with the specific language has a significant impact in the effectiveness of man-machine communication.

11. <u>User Behavior</u>

The more the language used as interface resembles the natural, the more comfort the user feels facing the system. There is though, a trade-off between user's convenience and the other aspects like ambiguity and complexity and consequently the cost of implementation. This of course does not exclude the case where an experienced user may use a "telegram" type dialect and does not feel very good when he must use natural language as it is used for conversational purposes.

12. <u>Cost of the Interface</u>

The design and implementation of natural language interfaces is in general more costly than any other interface. This cost is related to the degree of resemblance of the language used as interface to the natural language. As language becomes more restricted and formal, the cost of the interface decreases. The reason for the high cost of the natural language interface is related to the difficulty in making the machine understand the meaning of the human statement. In other words the ambiguity is the most significant factor in the cost of natural language interfaces. After all this, the use of such interfaces must be considered with care and there must very serious reasons to ignore the high cost associated with such a decision.

13. <u>Semantic Complexity</u>

From what we saw so far we may initially conclude that the implementation of natural language interfaces is very troublesome and is not worth consideration. This is not a general rule. There are cases where the use of natural language is suggested. All these cases concern the situation where the number of possible different messages which user wants to communicate is so big that the artificial languages have not the power to accomplish this task. Under these conditions a language interface which is adequately complex to handle the variety of messages is recommended.

There is a close relationship between the complexity of a program and the range and complexity of the interface to the program. This connection may often lead to the pitfall of "oversophisticated" interfaces as they are conceived by the user, and in this case the user may believe that the interface has unlimited capabilities and can answer any question.

English of course, is not the only "natural" language which may be used in man-machine interfaces. French, German or other languages may be candidates for the same purpose, but they should be expected to cause the same trouble . The reason is that certain grammatical characteristics like inflection of word endings, declensions and conjugation will only be recognized in context. Perhaps the invented languages like Esperanto would be better for the same purpose.

The main difficulty with any natural language is that it probably is not as logical as programming languages and this same attribute implies that artificial languages cannot be used for conversation.

B. FUNCTIONS PROVIDED AT THE INTERFACE

It is desirable that the number of functions provided will be as large as possible, so that the user will not be restricted during the interaction. The number of these functions characterizes the flexibility of the interface. Each of these functions plays its own important role for the effectiveness of communication between the man and the computer. In [Ref. 5:pp. 42], six categories of functions are distinguished:

1. Input/output Functions

They are absolutely necessary because they provide the actual means of man-machine interaction and for this reason they must be well designed and easy to understand. In the realization of the various dialogue categories two major aspects have to be considered. These are related in [Ref. 5:pp. 109] to the system's input and output.

a. Input considerations

For the input the designer must consider the following:

(1) The vocabulary. Having in mind all possible levels of user's experience, a recommended vocabulary for the novice user would resemble that utilized in the natural language. It will become thereafter, more and more restricted and stylized as user's experience with the system grows, ending in a dictionary for a highly professional dialect.

(2) Syntactical rules. The number must be as small as possible, so that they will easily be learned and memorized. They must also emphasize the accuracy and completeness of the entry.

(3) Format rules. They are in general undesirable. The user has already the mental load of memorizing syntactic rules and there is no reason why this load should be increased by having him memorize additional rules.

(4) Feedback. The system, must clarify any problems caused by incomplete or erroneous input and guide the user in locating the point of the problem.

b. Output considerations

The design principle for the output is that the system will provide the user with the relevant of information and in desirable form.

(1) The vocabulary. It should include natural or professional words. Such a dictionary is desirable to users of any level of experience. This is a necessary requirement to ensure that the output will be meaningful and understandable. Such a requirement cannot be satisfied by a vocabulary which is mostly composed by symbols, abbreviations and formal expressions.

(2) Output syntax and format. It must reflect the requirements of the actual output and the important parts must be easily recognized.

(3) Variety of representation. It must be possible that the output will be presented in a variety of forms so that the user will be able to select the form which fits to his/her needs best.

In this context, it is desirable that we shall identify another aspect. This is the need that the output will reflect facts and objects from input. This will enable the user to figure out how the results are related to the input data and how the information in hand has been produced.

2. Control Functions

They provide a mechanism for controlling the dialogue. They are separated in functions which are of interest to any user and the functions which address experienced users and and special tasks. Necessary control functions are those used to start , interrupt and terminate the dialogue.

3. Help Functions

They provide the user with special information about systems characteristics or the dialogue itself. An example of an interface which provides the user with adequate support, is that one in the Defense Data Network (DDN) [Ref. 6]. When the user wants to see which functions and commands are available he has just only to type:

```
HELP ?
```

and the system will provide him with a list of functions, commands and other features which are available. After this the user can request information about a command for example "LOGON" by typing:

```
HELP LOGON
```

and the system will provide him with information about how to "log on" the system.

4. Reorganization Functions

They enable switching from one dialogue category to another. This is a necessary feature for the interface. Although the primary concern of the design is the average user, different levels of experience will exist within the user community and, even more, a specific user's experience will continuously increase and he/she will probably want to be able to pass into more sophisticated levels of dialogue.

5. Communication Functions

The integration of communication functions in the system depends on the requirements of each particular application. Sometimes they are necessary, sometimes not. It is believed that they should be included in the interfaces of any contemporary information system. This comes from the fact that the requirements of the information systems in the 80's decade ask for integration of data processing and communications.

6. Extension Functions

They may be seen as a special category of control functions . They are suitable for use by experienced users in sophisticated tasks which are performed routinely.

C. DIALOGUE CATEGORIES

To organize the dialogue categories we should first distinguish several user categories in relation to their data processing experience. These categories are the novice, intermediate and experienced user. It is possible that all these categories will coexist within the same community. This is a result of the personnel turnaround which more or less is a problem for any organization. It is expected that an effective man-machine interface will cover the needs of users of any experience.

1. Menu-driven Dialogue

The system proposes task options to the user. The user can choose the option which he/she feels that fits better with his/her needs. This type of dialogue is appropriate for novice users.

2. System-initiated Requests

In this type, the system asks the user to make an input simultaneously providing the user with the necessary syntax for the input. If the input is not complete or correct with respect to the specification of the task, the system may respond in one of the following ways:

- Request for the user to make another input.

- Provide the user with the available alternatives to continue.
- Identify the incompletely or erroneously specified subjects.

This type of dialogue is appropriate for users of a certain level of experience with the specific system.

3. User-initiated Commands

Highly experienced users may prefer to specify their tasks in a highly stylized and formal fashion through the use of commands. System intervention is restricted in the specification of incomplete or erroneous information.

4. Natural Language Dialogue

This type of dialogue may be user-initiated or system-initiated. The vocabulary used resembles that which human beings use to communicate their ideas. The natural language dialogue imposes many implementation problems which increase when the user is the initiator of the dialogue and the degree of approximation to the natural language. Inexperienced users will probably desire such a dialogue. The difficulty of implementation makes it inappropriate for input, but suitable for guiding the user in the dialogue for the obvious reason that it is meaningful and easy to understand.

V. NATURAL LANGUAGE IMPLEMENTATION

A. THE UNDERSTANDING PROCESS

People want to communicate with the computer because they want the computer to do a task for them. In this task the computer may be "active", providing its services at the time of communication, or "passive" when it simply stores something for later use.

At the beginning, an idea is generated in the user's mind about the action he/she wants the computer to do. The understanding process has to be translated into a statement in a source language that the computer is able to understand. The statement expressed in source language is then entered into the computer. The computer must then "understand" the statement and do the appropriate actions. By "understanding" we mean that a certain program has to find which actions are being asked through the statement; this implies that the statement is meaningless for the computer if the computer cannot associate the statement with a set of actions.

This point is the beginning of all problems associated with natural language implementation and the reason is simple; it is very difficult to provide the computer with the capability to find which actions are related with a statement expressed in natural language. In other words it is difficult to make the computer understand the meaning of statements expressed in natural language. This liability of natural language statements to more than one interpretation is the so called ambiguity. It is the main reason which generates the requirement that in order for the natural language to be efficiently used as man-machine interface has to be "problem specific".

Although man-machine communication is a two-way process we will consider here only the direction from the user to the computer because we are interested in the design principles of a system in which the computer will understand user's statements and this is believed to be the primary issue of natural language implementation.

B. COMPONENTS OF A NATURAL LANGUAGE UNDERSTANDING SYSTEM

In studying the components of a natural language understanding system we have to distinguish two cases; understanding of single statements and understanding of text or dialogues.

1. Sentence Understanding

Sentence understanding is accomplished in three steps: the lexical analysis, syntactic analysis or parsing and semantic analysis [Ref. 7:pp. 41].

a. Lexical analysis

In this step, the statement is divided into its component words. Dividing a sentence this way is not difficult and it can be performed by scanning each sentence as it is in the input. The time required for this "scanning" is proportional to the length of the sentence. In this context, word is defined as a string of characters between two blanks. Of course when spoken language is used there is no way to see any words or spaces; in this case the lexical analysis requires information about the conditions under which the event occurs.

Under certain circumstances it is required that a word will be further subdivided into a "root" and a prefix or suffix. This kind of segmentation together with the rules about how the pieces have to be combined to form a word is called morphological analysis.

The list of words that can be recognized by a specific understanding program is contained in its dictionary or lexicon. Every understanding program has its own lexicon. The diversity of ways for word representation in the dictionaries from one program to another makes the use of the existing commercial dictionaries inefficient.

The selection of a lexicon requires two stages; first, the concepts which have to be expressed must be isolated, and second , the appropriate words which represent the concepts have to be selected. The isolation of the concepts can be done by looking at the actions that the program can do. In selecting the lexicon another requirement is that a distinction between a category of words in which addition of new entries is possible and a category of words where addition of new words seldom occurs, has to be made. Users are allowed to modify the arguments of certain actions , but not the actions by themselves. For example, in the statement

CALCULATE NET INCOMES FOR 1982

the word "1982" is a value for the "argument" year. User is allowed to change this value according to his/her desire.

The ambiguity in a natural language interface is related to how big is the subset of the natural language used at the interface. The use of the entire set of natural language creates such an ambiguity that implementation of the interface becomes infeasible.

b. Syntactic analysis

Natural language statements are prior to any analysis, simple strings of words, but they correspond to structured ideas. The purpose of the first step in the understanding process is to find a structure which corresponds to the structure of the meaning of the statement. The process of assigning a structure to an unstructured subject is called syntactic analysis or parsing. The inputs for which parsing is possible are considered to be "legal" whereas the rest are considered to be "illegal". We want an effective man-machine interface to provide us with as much flexibility in parsing as possible.

The parsing process makes use of a set of rules which govern the means by which high level structures can be generated from lower level ones. The collection of these rules is called grammar. The higher structure which corresponds to a specific complete grammatical statement is the sentence. The way in which statements are created by their components can be represented with a tree diagram which is called a parse tree. An example of a parse tree which demonstrates the formation of the statement from its components is shown in Figure 5.1.

There are two approaches to the parsing process: the top-down and bottom-up. In the top-down approach the parsing starts with the sentence and continues by hypothesizing the lower components. This hypothesization continues until actual words are needed. The words are then compared to the actual input. In the bottom-up approach parsing starts from the bottom with statement words and continues upwards. When the components of each intermediate constituent become available, the constituent is formed and the process continues until the top (sentence) has been formed.

Augmented transition network (ATN) [Ref. 8] is a very common approach for encoding the grammar. The network corresponds to the allowable transitions between the constituents of the statement. The arcs in the network guide the parser to take the appropriate actions during the parsing process in order to end up with the desired structure. An example of an ATN is provided in Figure 5.2.

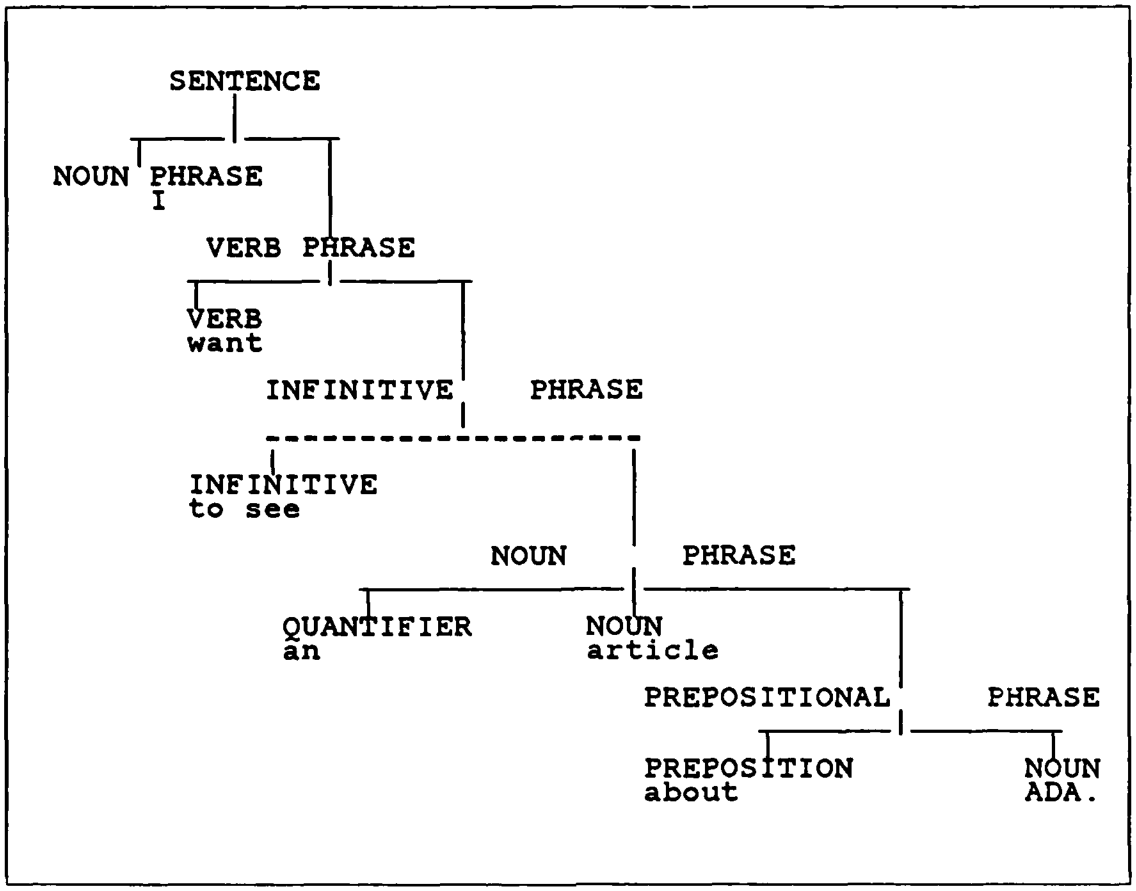


Figure 5.1 Parse Tree.

START --> noun phrase --> prepositional phrase --> END

Figure 5.2 Augmented Transition Network.

In addition to the lexical analysis, ambiguity plays an important role in parsing. In this case multi-word sequences of the same words are derived from the grammar. For example, the sentence

Show all the articles about software requirement analysis and design or maintenance

is ambiguous. It may be a request for articles about

software (requirement analysis and design) or maintenance.

It may be a request for articles about

software requirement analysis and (design or maintenance).

as well.

Ambiguity in parsing, implies that more than one analysis is required for the same sentence. Sager [Ref. 9], suggests the addition of restrictions concerning the acceptable word combinations, to help the final syntactic analysis. Another possible solution is that the ambiguities will be resolved in later stages or else the user has to be asked about what is intended. It is possible that the same language will have many different grammars. These grammars may assign different structures to the same statement This implies that their degree of usefulness for a particular understanding program varies.

c. Semantic analysis

It is the most important step in understanding process. In this step the statement is assigned a meaning. In other words the actions which the program has to take, are determined. The semantic analysis is accomplished in two stages; the semantic and pragmatic processing. In the semantic processing the meaning of a sentence is determined. In pragmatic processing the action which the target program has to take, is established. For example, in the statement

List all the customers in L. Angeles

we see that the meaning is related to the customers of L. Angeles. The action that the program has to do is to list those customers.

2. Text and Dialogue Understanding

In many cases understanding individual statements by a program is not enough. It is required that the system understands text. Several new problems emerge in this case.

The most important problem in dialogue understanding is that in the context of the dialogue, individual statements are very often incomplete. For example

```
User    : What is the length of USS Corry?
Computer: LENGTH 396 FT

User    : What is her draft?
Computer: DRAFT 19FT

User    : Her maximum speed?
Computer: SPEED 33 KNOTS
```

In this example the statements #2 and #3 are incomplete. The statement in step #2 makes use of the anaphoric pronoun "her" in order to refer to a subject which has already been mentioned in the first statement. In order for the system to understand the second question the anaphoric pronoun "her" must be tied to "USS Corry". In order for the third statement to be understood, the words which describe the information which the user looks for must be placed in the previous statement into the right slots. In this example this means that a query "what is her speed" will be constructed from the statement "her speed" and it will finally take the form "what is the maximum speed of USS Corry?"

The number of tasks which require natural language input which is greater than a sentence increases continuously. Many of these tasks require programs that can deal with informal and incoherent texts which are very difficult to understand particularly cases where idioms and metaphors are present.

There are several categories of programs and systems which deal with understanding of text and/or dialogues in natural language [Ref. 11].

a. Expectation-based parsing

The interpretation of a word meaning often leads to problems. A characteristic case is that one where the meaning of the word is associated with the concepts the word refers to. For example, in the statements

George got a cold
George got a book

the word "got" refers to two different conceptual structures and finding the exact meaning of "got" is very difficult.

The meaning of a word in such cases can be captured by use of rules which test the context of the sentence and decide about the meaning. In our case, when the word "got" is seen in the input, the test rules are activated and examine the rest of the sentence, expecting that probably the test may become true. In the opposite case an action is taken which assigns a structure to the statement. These rules can be prepared so as to cover any possible meaning of the word "got" or any other word which creates the same problem.

Conceptual analyzers are tools which assist in the disambiguation of such words. The process of disambiguation consists of predicting likely meanings and picking up from the word's definition the sense that corresponds better to the predictions.

b. Script-driven parsing

In this case, conclusions about a story are generated on the basis of an internal description of the story, a kind of "script", consisting of basic elements that normally occur in events of the same type as that one being reported. For example a script for a battle will have slots for the time, place, damages and victims of the battle. Any story about some battle will then be treated as an instance of the script. It can be noticed here that what changes is only the value of the slots. For example in the sentences

The battle of El Alamein in 1943 had 9645 deaths

and

The battle of Rhein river in 1942 had 11233 deaths

are instances of the same script. What changes is the value of place, time and victims slots.

A tool which operates on this principle is the FRUMP conceptual analyzer [Ref. 10:pp. 42]. FRUMP scans the first sentence of the text in input, looking for words or phrases whose meanings are referred in one of the scripts in his dictionary. When a script is picked up, attempts to fill up the script slots are made. These attempts are based on rules which determine where to look in the input text for words or phrases whose meanings might fill up the slot. FRUMP automatically ignores things that cannot fit into the script.

c. Interest-driven parsing

In this approach the principle of script-driven parsing is maintained and is combined with the use of frames. Frames are smaller than scripts but more modular and are integrated with each other. The idea in the interest driven parsing is that the more interesting frames when they refered to the sentence become the driving mechanism for understanding the sentence. For example, in a sentence begining like

"A long duration earthquake........"

the word "earthquake" will start several actions. First, the meaning of the word "earthquake" will be examined. A definition of earthquake wants the word as the primary participant in a frame for natural disasters. This frame describes not only what an earthquake is, but also its cause and its results. The analysis of the sentence is then governed further by the frame for natural disasters which looks for pieces to fill the existing slots in the frame. Preceeding and following inputs which were not so interesting as "earthquake" are now candidates to be selected for filling up the slots. For example "tectonic" is picked up to fill the slot for the type of earthquake which participates in the event of natural disaster. In addition to this, predictions about the rest of the sentence may be done. The word "leveled" in the sentence

A long duration tectonic earthquake leveled the
city of San Fransisco

is interpreted to mean "destroyed" because that is a reasonable prediction of what a long duration tectonic earthquake can cause. As the analysis progresses other frames may be picked up, because one piece of input may be referred in an interesting frame which is not currently active. The analysis is then governed by these frames.

A tool used for parsing based on interest-driven analysis, is the Integrated Partial Parsing (IPP) system [Ref. 10:pp. 44]. A particular characteristic of the IPP system is that it ignores some words like "be", "have", "give", "get" and so on, because they are referred to low interest frames. This characteristic becomes a problem when low interest frames are referred by words of the text but the text is of great interest to us.

d. Goal-driven parsing

The analysis of the text is performed in bottom-up fashion, using expectations. These expectations are found by lexical analysis of the words in the sentence. The system considers only the most recent expectations. This has two consequences. First, the number of the things that must be cheked is small, and second, failure of one expectation can block other expectations.

A system which is based on this principle is McMAP [Ref. 10:pp. 46]. It can handle several different texts with multiple sentences, by keeping track of all structures constructed so far and by using inference rules to tie all these structures together. The main advantage of McMAP is that it can check the reasonal -y of a given interpretation.

VI. THE DESIGN OF NATURAL LANGUAGE INTERFACES

To design an acceptable natural language interface, we have to consider several crucial aspects:

- The interface must be application specific.
- The subset of the natural language which is going to be used, must be sufficient to support the particular application.
- The interface must be to some degree transportable.

The size of the natural language subset is a very important factor in the definition of the design approach. We distinguish three cases:

- The number of statements which are needed to be entered for the accomplishment of a specific task is small may be implemented through a finite number of options presented to the user on a screen.
- The number of the statements which have to be entered at once is large enough and they cannot be efficiently combined, because of the large number of possible combinations, to create a finite number of options to the user, but the subset used is small.
- The number of statements which have to be entered at once is so large (text or complex sentences which have pronouns or are referenced between them) that in order for the system to be able to understand the meaning of the input, it must capture as many characteristics of the language, as possible.

Another factor which has to be considered, is the application environment. We distinguish four application environments:

- Data retrieval
- Natural language dialogues and text processing.

* Text knowledge systems.
* Natural language programming.

A. DESIGN APPROACHES

1. Menu-driven Natural Language Dialogue

This approach is indicated when the total number of related statements which will probably be entered is small. This allows the presentation of various options to the user, who in turn selects between them and constructs the statement as a combination of these options.

In a possible implementation of this approach, the options are presented through a screen, as in Figure 6.1.

```
COMMANDS: List   Show   List all   Show all   Show the
=======================================================

FEATURES     CONNECTORS     QUALIFIERS     QUANTIFIERS
class        and            built by       a
name         of                            all
speed        for                           number
fuel         in
crew         at

COMPARISONS            NOUNS                ATTRIBUTES
smaller than           ship                 <name     >
greter than            station              <station  >
equal to               mission              <class    >
equivalent to          weapons              <weapons  >
different from                              <number   >
========================================================

QUERY STATUS : List  all ships of class DDG ...
```

Figure 6.1 Menu-driven Natural Language Dialogue.

Rich [Ref. 8], calls this approach "Window method" considering each screen presented to the user, as composed by windows of information.

In the screen of figure 6.1 three windows exist. The user starts from the window which is first activated by the system. After the selection of one of the available words, the next window is activated, in order for the user to proceed with the next selection. Two things are important after each selection:

- The screen must be updated with respect to the progress of the dialogue so far.
- The windows which contain options for the next steps will be modified to present only the options which are are necessary for the user to proceed further.

The first action is related to the objective for adequate feedback to the user with respect to the progress of dialogue. The second , is related to the objective that the user must be provided with just the information he/she needs to continue the dialogue and nothing more.

It is believed that in order for such a screen to address user's needs, it should contain <u>at</u> <u>least</u> the following windows:

- Command window: It should contain command options for completing the statement, for interrupting and/or terminating the dialogue, help and control commands.
- Word window: It must contain any word which is expected to be used in the composition of some statement, except commands.
- Dialogue progress window: It should contain any information concerning progress of the dialogue so far, for example the part of the statement which has already been created.

The implementation of this approach requires the following components:

- Lexicon or dictionary or vocabulary: It contains the words which have to be recognized by the system.

- Grammar: It contains the rules that govern the formation of the statement, from the words which are presented on the screen.

In this approach, lexical analysis is not complex. The reason is that every possible constituent of input already exists and is provided automatically by the system. Parsing is performed as user's statement is entered and includes the building of the parse and updating the window. Semantic processing is easy and is performed while the parse tree is being built up.

A very important characteristic of this method is that any statement composed of words which are provided as options will be <u>any way</u> correct and ready for syntactic and semantic analysis and the possibility of rejection by the system exists. Another important characteristic is that the user is in some way restricted to move onto a small number of predefined paths, but this is the trade-off for the error-free dialogue.

This approach is suitable for cases where the semantic complexity of the domain is low and the number of the options which have to be provided is small.

2. <u>User-initiated Dialogue with Simple Statements</u>

This approach is indicated when the number of statements which have to be constructed is large and the subset of natural language used is small. If the number of statements is large beyond the point that allows the system to maintain the dialogue by itself, then the previous method is not indicated.

In this approach the user constructs entire statements, which are then entered into the system. The system is assigned the responsibility of understanding each statement and implementing the required actions. The idea behind this approach is the combination of parsing and semantic analysis in a single step. So, the understanding is accomplished in two steps. The method implements a kind of grammar which contains rules for both parsing and semantic analysis. Two requirements govern the design of these rules. First, the rules are designed in such a way that the statements which can be successfully parsed correspond closely to the actions of the target program. Second, the structure, which is assigned by the parser to the statement, also corresponds to the target program. A problem which may appear with this approach, is that some inputs, which do not correspond closely enough to the target program, may be accepted. This occurs if the rules of the grammar are not sufficiently constrained. Another problem is that a correct statement may not be parsed if it is not associated with any of the specific sets of actions of the target program. This is a result of the situation where the rules of the grammar are associated to some set of actions. An example of a parse tree generated by such rules is shown in Figure 6.2.

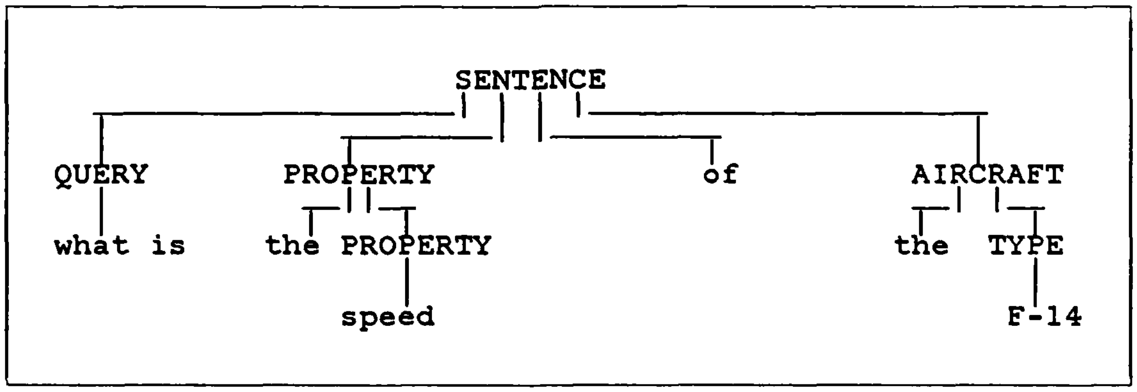


Figure 6.2 Syntactic Parse Tree.

In this example the categories "aircaft" and "type" are designed to correspond to the target set of actions which obviously concerns facts and actions about aircrafts.

Here the similarity between the structure of the query and the parse tree can be noticed, which makes the production of the query easy.

This method is suitable when the subset of the natural language is small. It becomes inappropriate when this subset is so big that some of the characteristcs of the language must be used in the understanding process. As the rules are expanded to deal with larger pieces of language, they become so complex that the implementation of the method becomes infeasible.

3. User-initiated Dialogue with Complex Statements

When a large part of the natural language is used as interface it is necessary to capture as many characteristics of the language as possible in the rules used to understand it. A grammar which satisfies such a requirement maps structures of statements onto structures of target program actions. When both the assignement of a structure to the statement and the translation of the structure to target actions are complex then the understanding must be accomplished into two steps. First, rules which allow the parser to assign structures, which capture various characteristics of the language of the statement are used. Second, other rules are needed for the translation of the structures to actions.

The difference from the previous approach is that two separate sets of rules are needed for understanding the input which is performed in two separate steps.

The problem with understanding of complex sentences, is that regardless of the parsing strategy the components which will finally be used are very often difficult to determine in advance. The final choice of these constituents is a search process which may make use of various techniques, like those described in the previous chapter, to solve the problem.

To minimize the number of intermediate constituents which may be at some stage rejected, most parsing rules are augmented with tests that must be satisfied in order for the rules to activate. Some of these checks test for syntactic properties, others for semantic and yet others for specific properties.

There are two ways to perform the third step, the semantic analysis. The first way is to inter-leave syntactic and semantic processing and associate with each grammar rule a set of actions which will be applied whenever the rule is applied. There are two reasons why this technique is very often inappropriate:

- The need for backtracking cannot be satisfied once some actions have already been associated with the rules.
- The objective of interface transportability cannot be accomplished because the parser and the grammar are difficult to be transferred in many applications.

A second technique separates completely the parsing from the semantic processing and satisfies both these requirements.

B. INTERFACE ARCHITECTURE

We have seen so far, that the size of the natural language subset is a very important factor for the definition of the design approach. We have seen also that not all of the components of a natural language understanding system are equally important to each design approach and that it is also possible that a combination of the components will exist.

1. Interface Requirements

In our attempt to define an architecture for the natural language interface, we shall first look at the requirements which that interface must satisfy. The <u>minimum</u> requirements which the interface should satisfy, are listed below:

- Flexible parsing: User's input must be acceptable when minor linguistic errors exist. In addition, the user must be provided with adequate feedback relative to the cause of the errors and even more he/she must be provided with recommendations about the alternatives available for continuing the dialogue.
- Adequate feedback: The user must be informed as often as possible about the actual state of the dialogue and what the system has understood.
- Intension recognition: The system must be able to distinguish the meaning of similar expressions used within different contexts. This is a very important requirement for text understanding systems.

These <u>minimum requirements</u> have to be fulfilled in order for the interface to be acceptable, but in order for the interface to be graceful too, the following <u>additional requirements</u> must be satisfied:

- Transportability: The interface must be in some extend "application-independent".
- Modifiability: User must be allowed to modify the components of the interface for example the lexicon, rules, etc. This allows the addition of new entries (knowledge acquisition) and/or improvement of its features.

A natural language interface which has the capabilities of satisfying these requirements, addresses most of the objectives of the man-machine interface. There are, though, some objectives which the natural language interface does not address. These are the reliability and transparency of the dialogue. These objectives are very difficult to be realized by natural language interfaces. The primary reason is the ambiguity in the natural language which exposes the user to many errors (unreliable dialogue) and he/she also meets difficulties in predicting system's behavior (lack of transparency).

2. Interface Characteristics

Before examining the tools, which a natural language interface must make use of and its functions as well, we must decide about the scope of the interface. Two options are available:

- The interface will be of limited scope; this means that it will be used for implementing one of the approaches.
- The interface will be of broad scope; this means that it will be used for implementing more than one of the approaches.

The advantage of a general purpose interface is that it addresses a larger number of the user's characteristics and information needs. There is, though, a trade-off between the breadth of scope and the complexity and scope of the interface. The broader is the scope, the higher is the complexity and the cost. A restricted purpose interface addresses a smaller range of the user's characteristics and information needs, but favors lower complexity and cost.

The interface which is proposed is a general purpose interface with the following important characteristics:

- It combines all the three design approaches.
- It provides the capability of modifying the content of system's components.
- It is transportable.

Several implications come out of these characteristics and they are listed below:

- The interface may be used for system-initiated dialogue, simple statement understanding and complex statement/text understanding.

- User's conceptual view is separated from the details of the particular application.
- Not all of the components will be important for any case of input understanding.

A general overview of a transportable natural language interface for data retrieval is presented in [Ref. 4]. This interface is the basis for the general purpose interface presented in figure 6.3.

The new features which have been added to the interface are the following:

- Complete separation of the natural language portion from the application portion.
- A means of cooperation between the natural language part and the application part.
- A means for extending the existing knowledge bases.
- A means for extending or modifying the sets of rules.

3. Description of the Proposed Interface

The interface consists of two portions; the natural language portion and the translation portion. Their joint task is to process any natural language input, to understand it and generate a kind of code which represents the meaning of that input and consequently the interpretation of user's intentions.

The user can select one of the following options as far as the type of dialogue is concerned:

- System-initiated natural language dialogue.
- User-entered simple statements.
- User-entered complex statements and/or text.

By specifying his/her option via menu selection, the dialogue controller activates the components which are necessary for the realization of the dialogue. Then, one of the three understanding approaches is applied. Given that text understanding is the most difficult case because of the

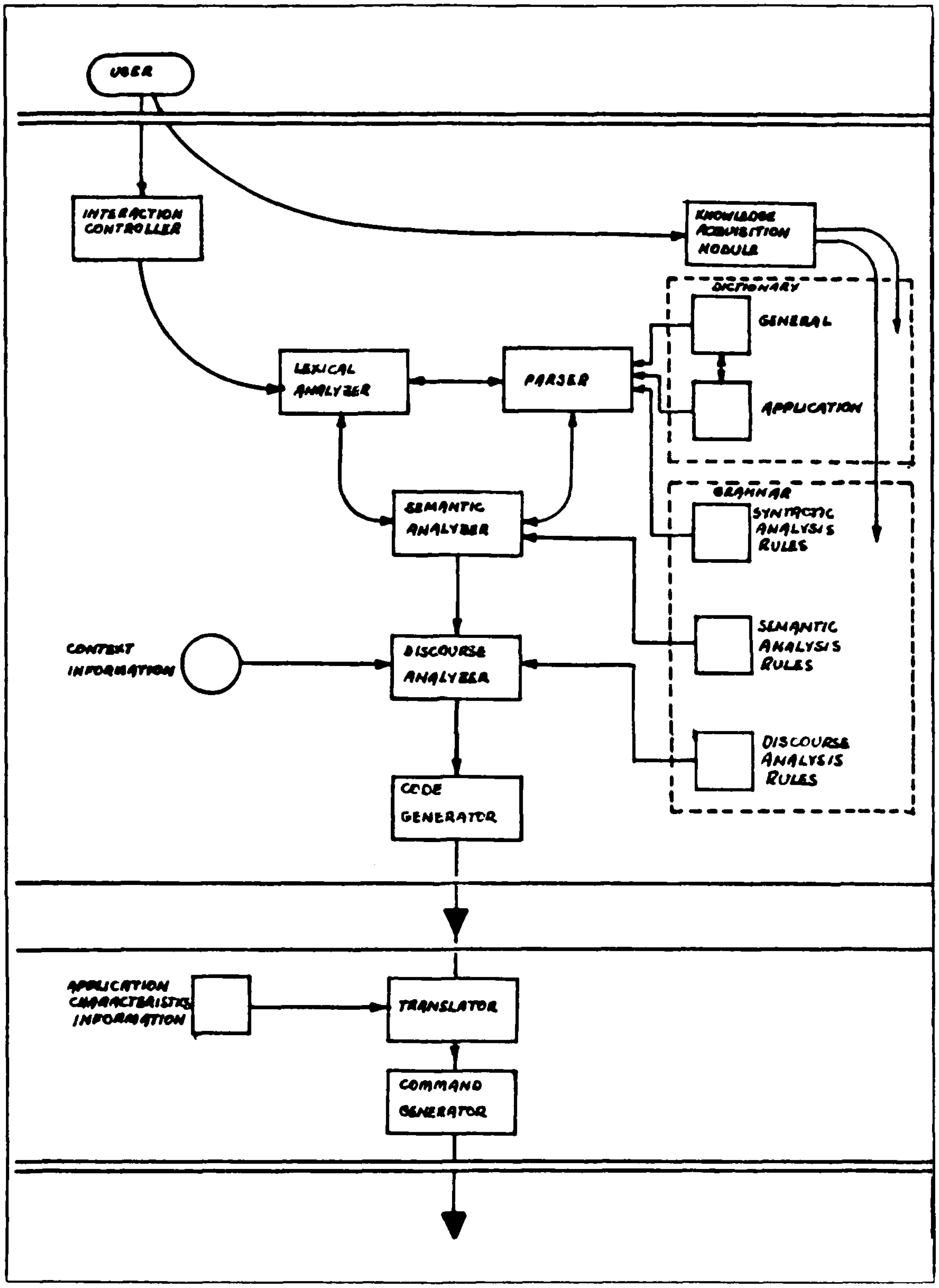


Figure 6.3 Diagram of the Natural Language Interface.

increased ambiguity, it was found appropriate that a special module for the disambiguation of the input would be extremely useful in solving the problem, by filling the product of semantic analysis with additional linguistic information and resolving various types of references.

The next step includes the transformation of a complete interpretation of the input to some kind of code (meaning representation code). This code is ready to be translated in the application translation module, into a representation which makes explicit reference to the characteristics and structure of the specific application. The translator, using facts from user's conceptual view and information about application characteristics like files, records, fields etc in case of database, takes the meaning representation code and produces an application-specific code. This code consists of various commands which are accepted by the application control system, for example the database management system.

4. Tools

Several tools are employed at the various stages of processing the natural language input at the interface:

- Application dictionary: It consists of words which are related to the application (domain specific).
- General dictionary: It contains words which are not directly related to the application (domain independent).
- Syntactic rules: Rules used for the syntactic analysis (parsing).
- Semantic rules : Rules used for semantic analysis.
- Context information: Information referenced in case :
  - The semantic analyzer needs information about context to assign meaning to a specific statement.

  - Complex or interrelated statements or text are entered .

- Discourse analysis rules: Rules which determine the way in which context in formation will be used.

Another important module which was left to be described at the end because it is not directly retated with the dialogue, is the knowledge acquisition module. This module allows the user to extend the domain of the interface by creating new entries and rules or modifying the domain by changing entries and rules.

5. <u>Parsing and Understanding with Word Experts</u>

This approach makes use of a model of natural language understanding, called Word Expert Parser [Ref. 10:pp. 89]. Under this approach each word is treated as a complex procedural knowledge source which contains decision paths, which reflect the range of knowledge about a specific word, necessary to understand that word in various contexts. The idea behind this approach is that a language can be understood through a controlled exchange of meaningful signals and concepts among internally complex experts.

The Word Expert Parser (WEP), is organized in three levels:

- Top : it represents the control environment where the experts run.
- Middle :it consists of the protocol for the inter-expert communication.
- Low : it concerns the intra-expert structure of the expert's decision logic.

Considering the general model environment, there is one expert for each morpheme (syntacticaly or semanticaly meaningful lexical unit which cannot be further subdivided). During lexical analysis, each sentence is morphologicaly analyzed to identify the relevant word experts which then are manipulated by the parser. Control of parsing is first assigned to the leftmost expert which, thereafter, takes control of the model. The other words are read "on demand" by other experts or after request from the top level.

The control environment, where the experts run, contains the following features:

- memory context : results of understanding processes of previous sentences.
- concepts : data objects which represent understanding thoughts in various stages of refinement and which become available to running experts when they do appropriate questions.
- parser state : description of the current focus of the parser, as it is described by previous running experts.
- signals : information about parser nature.
- demons : mechanisms which watch computational results of the running experts and reactivate other experts when required or after a given period of time has been passed.

a. Running a Word Expert

Experts are activated in three cases:

- Questions are addressed to them by other experts.
- They must respond to newly created concepts or signals.
- They become the next word to be read in the sentence.

Four parameters characterize the reasons of expert awakening:

- Reentry points : inform experts where decision logic has to be restarted.
- Signals : indicate the nature of expert awakening.
- Concepts : accompany various experts.
- Reference : reference to another expert if awakening has occurred in response to a prior request.

b. Word Expert behavior

The behavior of a word expert is expressed through primitives which establish the range of that behavior. Several such primitives [Ref. 10:pp. 97], are suggested:

- Request for the next word of the sentence which must be read.
- Information request from other experts.
- New information reporting.
- Building new concepts or contributing to existing concepts.
- Information requests about concepts.
- Suspending running experts to await future reawakening by time-out demons.
- Terminating expert running after identifying word's role in the sentence.

These primitives are organized in two sets; the Lexical Interaction Language (LIL) which contains primitives for inter-expert communication and the Sensed Discrimination Language (SDL) which contains experts concerning expert's decision logic. Both sets of primitives and their classification are illustrated on Table II.

c. Sentence understanding with Word Expert Parsing

In this part, the implementation of the WEP approach on the sentence "A ship carrying missiles", is provided. The entire process is accomplished in 21 steps as it is described below:

- The parser reads the word "A", retrieves its relative expert from the memory and starts running it.
- The expert of word "A" is being prepared to accept the upcoming concept (ship).
- Initial construction of the "ship" expert.

TABLE II

Word Expert Behavior Primitives

LEXICAL INTERACTION LANGUAGE PRIMITIVES

Posting Restarting Demons

AWAIT CONCEPT, AWAIT SIGNAL, AWAIT WORD

Look-Ahead Functions

PEEKW, READW

Message Sending

SIGNAL, REPORT

SENSE DISCRIMINATION LANGUAGE PRIMITIVES

Linquistic Group Structuring

LITERAL, OPENG, DECLAREG, BREAKG, LINK, CLOSEG, IDIOM

Concept Building and Augmenting

BUILDC, REFINEC, ROLEC, ASPECTC, STOREC

Control Flow

NEXT, CONTINUE, PAUSE, ALIAS

Memory Interaction

VIEW, BINDC

- Running of the suspended expert "A" is resumed, after the construction of the concept which refers to the expert "ship" has been reported.
- Initial execution of the "ing" expert.

- The "ing" expert is being prepared to accept a response about the main verb (carry).
- Initial execution of the "carry" expert which then branches to an action node in order to be tied with the "ing" expert.
- Execution of the "carry" is resumed.
- Word expert for "ing" resumes its third execution, to give the "ing" expert the name of the structure being built by "carry".
- The "s" expert initiates a new sequence.
- The "ing" expert branches to a node to wait for the upcoming concept.
- The "s" expert continues awaiting the same concept awaited by "ing".
- The "missile" expert discriminates among possible word senses.
- The "s" expert opens new lexical group changing state of the parser to "action construction".
- The "missile" expert resumes execution reporting the concept structure it has created initially.
- The "s" expert resumes execution using the desired concept as input.
- The "ing" expert resumes execution to sort out the meaning of the text fragment for which it has coordinated the parsing process.
- The "carry" expert resumes execution to process the concept structure sent by the "ing" expert.
- The "carry" expert resumes execution refining the meaning of its action.
- The "ing" expert resumes for final execution. The carrying action and its object are stored in the active memory.
- Execution of the "." expert, announcing the end of the sentence and completion of analysis.

```
                    EXPERT1 "A"

ENTRY  : INITENTRY          SIGNAL  : *BREAK*
EXPERT : *WEP*              CONCEPT : NIL

  INITENTRY
    ENTRY0
    N0/A      OPENG        SENDING
                           *ENTITY-CONSTRUCTION*
                           SIGNAL
              DECLAREG     POSTING SIGNAL DEMON
                           RESUME ON FIRING AT :
                           ENTRY1


                    EXPERT1 "A"

ENTRY : ENTRY1             SIGNAL  : *PAUSE*
EXPERT : *WEP*              CONCEPT : NIL

  ENTRY1
    N0/A      AWAIT        POSTING CONCEPT DEMON
                           RESUME ON FIRING AT :
                           ENTRY2
                           RESUME ON EXPIRATION
                           AT : ENTRY3


                    EXPERT4 "ship"

ENTRY : INITENTRY          SIGNAL  :
                           *ENTITY-CONSTRUCTION*
EXPERT : EXPERT1           CONCEPT : NIL

  INITENTRY
    ENTRY0
    N0/Q      SIGNAL       BRANCHING ON
                           *ENTITY-CONSTRUCTION*
    N3/A      DECLAREG
```

```
                CONTINUE
     ENTRY1
     N0/A       CREATEC      CREATING C#CONCEPT7:
                             ENTITY
                REFINEC      REFINING C#CONCEPT7:
                             C#L#SHIP
                REFINEC      REFINING C#CONCEPT7:
                             C#DDG-SSBN
                LINK
                CLOSEG       SENDING
                             *COMPLETE-ENTITY*
                             SIGNAL
     N1/A       REPORT       REPORTING C#CONCEPT7


                  EXPERT1 "A"

 ENTRY : ENTRY2              SIGNAL : NIL
 EXPERT : EXPERT4            CONCEPT :C#CONCEPT9

   ENTRY2
     N0/A       BINDC        BINDING IN
                             FOCUS AREA FAILS
     N2/Q       BOUND        CONCEPT2 VARIABLE
                             UNBOUND
     N4/A                    REPORT C#CONCEPT7


                  EXPERT8 "ing"

 ENTRY : INITENTRY          SIGNAL :
                             *COMPLETE-ENTITY*
 EXPERT : EXPERT4            CONCEPT : NIL

   INITENTRY
     ENTRY0
     N0/Q       SIGNAL       BRANCHING ON
                             *COMPLETE ENTITY*
```

```
    N3/A       OPENG        SENDING
                            *ACTION-CONSTRUCTION*
                            SIGNAL
               DECLAREG     POSTING SIGNAL DEMON
                           RESUME ON FIRING AT:
                            ENTRY1


                 EXPERT8 : "ing"

ENTRY : ENTRY1              SIGNAL : *PAUSE
EXPERT : *WEP*              CONCEPT : NIL

  ENTRY1
    NO/A       BUILDC       CREATING C#CONCEPT12:
                            ENTITY
                            C#CONCEPT12ROLE:
                            AGENT
               BINDC        BINDING IN
                            ACTIVE AREA SUCCEEDS
               READW
               SIGNAL       SENDING *ING* SIGNAL
                            TO: EXPERT9
               AWAIT        POSTING SIGNAL DEMON
                            RESUME ON FIRING AT:
                            ENTRY2
                            RESUME ON EXPIRATION
                            AT: ENTRY3


                 EXPERT9 "carry"

ENTRY : INITENTRY           SIGNAL : "ING"
EXPERT : EXPERT8            CONCEPT : NIL
 INITENTRY
   ENTRY0
   NO/Q        SIGNAL       BRANCHING ON "ING"
```

```
N4/A      BINDINPUT
          DECLAREG
          ALIAS
          CONTINUE
ENTRY1
NO/A      CREATEC      CREATING C#CONCEPT7:
                       ACTION
          REFINEC      REFINING C#CONCEPT7:
                       C#L#CARRY
          REFINEC      REFINING C#CONCEPT7:
                       C#CARRY
          LINK
          CLOSEG       SENDING
                       *COMPLETE-ACTION*
                       SIGNAL
          PAUSE        POSTING SIGNAL DEMON
                       RESUME ON FIRING AT:
                       ENTRY4


                EXPERT9 "carry"

ENTRY : ENTRY4                 SIGNAL : *PAUSE*
EXPERT : *WEP*                 CONCEPT : NIL

  ENTRY7
    NO/A      SIGNAL       SENDING "ING" SIGNAL
                           TO: EXPERT8
                           WITH: C#CONCEPT7
              AWAIT        POSTING SIGNAL DEMON
                           RESUME ON FIRING AT:
                           ENTRY3
                           NO RESUMPTION ON
                           EXPIRATION
              AWAIT        POSTING SIGNAL DEMON
                           RESUME ON FIRING AT:
```

```
                                         ENTRY5
                                         NO RESUMPTION ON
                                         EXPIRATION


                         EXPERT8 "ing"

ENTRY : ENTRY2                           SIGNAL : "ING"
EXPERT : EXPERT11                        CONCEPT : C#CONCEPT7
 ENTRY2
   NO/A         AWAIT                    POSTING SIGNAL DEMON
                                         RESUME ON FIRING AT:
                                         ENTRY4
                                         NO RESUMPTION ON
                                         EXPIRATION


                         EXPERT21 "s"

ENTRY : INITENTRY                        SIGNAL :
                                         *COMPLETE-ACTION*
EXPERT : EXPERT9                         CONCEPT : NIL

  INITENTRY
    ENTRY0
    NO/Q        SIGNAL                   BRANCHING ON
                                         *COMPLETE-ACTION*
    N2/A        OPENG                    SENDING
                                         *ENTITY-CONSTRUCTION*
                                         SIGNAL
                DECLAREG
                PAUSE                    POSTING SIGNAL DEMON
                                         RESUME ON FIRING AT:
                                         ENTRY2


                         EXPERT8 "ing"
```

```
ENTRY : ENTRY4               SIGNAL :
                             *ENTITY-CONSTRUCTION*
EXPERT : EXPERT21            CONCEPT : NIL

  ENTRY4
    NO/Q      SIGNAL         BRANCHING ON
                             *ENTITY-CONSTRUCTION*
    N1/A      AWAIT          POSTING CONCEPT DEMON
                             RESUME ON FIRING AT:
                             ENTRY5
                             NO RESUMPTION ON
                             EXPIRATION


                   EXPERT21 "s"

ENTRY : ENTRY2               SIGNAL : PAUSE
EXPERT : *WEP*               CONCEPT : NIL

  ENTRY2
    NO/A      AWAIT          POSTING CONCEPT DEMON
                             RESUME ON FIRING AT:
                             ENTRY1
                             NO RESUMPTION ON
                             EXPIRATION


                   EXPERT22 "missile"

ENTRY : INITENTRY            SIGNAL:
                             *ENTITY-CONSTRUCTION*
EXPERT : EXPERT21            CONCEPT: NIL

  INITENTRY
    ENTRYO
    NO/Q      SIGNAL         BRANCHING ON
                             *ENTITY-CONSTRUCTION*
    N3/A      CREATEC        CREATING:
```

```
                          C#CONCEPT26: ENTITY
              REFINEC     REFINING:
                          C#CONCEPT26:
                          C#L#SHIP
              DECLAREG
              LINK
              CLOSEG      SENDING
                          *COMPLEX-ENTITY*
                          SIGNAL TO: EXPERT30
              BUILDC      CREATING
                          C#CONCEPT38 ROLE: NIL
              AWAIT       POSTING CONCEPT DEMON
                          RESUME ON FIRING AT:
                          ENTRY2
                          RESUME ON EXPIRATION
                          AT: ENTRY3


                EXPERT30 "s"

ENTRY : INITENTRY         SIGNAL :
                          *COMPLEX-ENTITY*
EXPERT : EXPERT23         CONCEPT : NIL

  INITENTRY
    ENTRY0
    NO/Q       SIGNAL     BRANCHING ON
                          *COMPLEX-ENTITY*
    N4/A      OPENG       SENDING
                          *ACTION-CONSTRUCTION*
                          SIGNAL
              DECLAREG


                EXPERT22 "ship"

ENTRY : ENTRY3            SIGNAL : *ELSE*
```

```
EXPERT : *WEP*              CONCEPT : NIL

  ENTRY3
    NO/A       REPORT       REPORTING C#CONCEPT26


                 EXPERT21 "s"

ENTRY : ENTRY1              SIGNAL : NIL
EXPERT : EXPERT23           CONCEPT : C#CONCEPT26

  ENTRY1
    NO/A       REFINEC      REFINING C#CONCEPT26
               REPORT       REPORTING C#CONCEPT26


                 EXPERT8 "ing"

ENTRY : ENTRY5              SIGNAL : NIL
EXPERT : EXPERT21           CONCEPT : C#CONCEPT26

  ENTRY5
    NO/A       BUILDC       CREATING C#CONCEPT33:
                            ACTION
                            C#CONCEPT33 ROLE:
                            NIL
               BUILDC       CREATING C#CONCEPT34:
                            ACTION
                            C#CONCEPT34 ROLE:
                            NIL
               ASPECTC
               ASPECTC
               BUILDC       CREATING C#CONCEPT35:
                            ACTION
                            C#CONCEPT35 ROLE:
                            NIL
               BINDC        BINDING IN PLAUSIBLE
                            AREA SUCCEEDS
```

```
N1/Q           BOUNDC       CONCEPT6 VARIABLE
                            BOUND TO
                            C#CONCEPT33
    N2/A       BUILDC       CREATING C#CONCEPT36:
                            ENTITY
                            C#CONCEPT 36 ROLE:
                            NIL
               BUILDC       CREATING C#CONCEPT37:
                            ENTITY
                            C#CONCEPT37 ROLE:
                            NIL
               BINDC        BINDING TO ASPECT OF
                            C#CONCEPT33 SUCCEEDS
               BINDC        BINDING TO ASPECT OF
                            C#CONCEPT33 SUCCEEDS
               SIGNAL       SENDING *AGENT* SIGNAL
                            TO: EXPERT9
                            WITH: C#CONCEPT9
               SIGNAL       SENDING *OBJECT* SIGNAL
                            TO: EXPERT9
                            WITH: C#CONCEPT26
               AWAIT        POSTING CONCEPT DEMON
                            RESUME ON FIRING AT:
                            ENTRY6
                            NO RESUMPTION OF
                            EXPIRATION


                  EXPERT9 "carry"

ENTRY : ENTRY3              SIGNAL : AGENT
EXPERT : EXPERT10           CONCEPT : C#CONCEPT9

  ENTRY3
    NO/A        ASPECTC
               ROLEC        C#CONCEPT9 ROLE:
```

```
                                    AGENT
        N1/Q        VIEW            EXAMINING CONCEPT:
                                    C#CONCEPT9
                                    POSSIBLE VIEWS:
                                    C#PERSON
                                    C#OBJECT
                                    C#SYSTEM
                                    CHOICE: C#SYSTEM
         N4/A        REFINEC        REFINING C#CONCEPT13:


                            EXPERT9 "carry"

    ENTRY : ENTRY5                  SIGNAL : OBJECT
    EXPERT : EXPERT8                CONCEPT : C#CONCEPT26

      ENTRY5
        N0/A        ASPECTC
                    ROLEC           C#CONCEPT28 ROLE:
                                    OBJECT
        N1/Q        VIEW            EXAMINING CONCEPT:
                                    C#CONCEPT26
                                    POSSIBLE VIEWS:
                                    C#ANYTHING
                                    C#WEAPONS
                                    C#PEOPLE
                                    CHOICE: C#WEAPONS
         N2/A        REFINEC        REFINING C#CONCEPT13
                                    C#CARRY-WEAPONS
                     REPORT         REPORTING C#CONCEPT13

                            EXPERT8 "ing"

    ENTRY : ENTRY6                  SIGNAL : NIL
    EXPERT : EXPERT9                CONCEPT : C#CONCEPT13

      ENTRY6
        N0/A        STOREC
```

```
                STOREC
                REPORT      REPORTING C#CONCEPT9


                 EXPERT49 "*per*"
         ENTRY : INITENTRY          SIGNAL :
                                    *COMPLETE-ENTITY*
         EXPERT : EXPERT22          CONCEPT : NIL
           INITENTRY
             ENTRY0
             NO/A       BREAKG      SENDING
                                    *BREAK* SIGNAL
```

6. Interface Operation Algorithms

Of great interest for the implementation of the natural language interface are the algorithms which describe the function of the natural language understanding portion. That portion will be represented in two levels of detail; the organization level and the functional level. The first concerns the organization of the interface to implement the three approaches and the second the components and functions of the system, in each case.

Table III shows an algorithm written in pseudocode, describing the function of the interface at the organization level. In this algorithm the following modules are used to represent the three approaches:

- Menu-driven dialogue module (MDD).
- Single sentence dialogue (SSD).
- Complex sentence dialogue (CSD).

Menu-driven dialogue is described in algorithm of Table IV and flowchart of Figure 6.4 . The following modules are being used:

- UPD_STATUS : updates status of dialogue.

TABLE III

Interface Organization Algorithm

```
ALGORITHM NLI_ORGANIZATION

Begin
read selection
if selection = MMD then
begin
  parser = active
  semantic analyser = active
  do MMD
  else if selection = SSD
  begin
    lexical analyzer = active
    parser = active
    semantic analyzer = active
    do SSD
    else if selection = CSD
    begin
      lexical analyzer = active
      parser = active
      semantic analyzer = active
      discourse analyzer = active
      do CSD
      else
      do E_INTERRUPT
    end if
  end if
end if
END
```

- SYN/SEM_ANALYSIS : performs syntactic and semantic analysis.
- E_INTERRUPT : interrupts and notifies user if input is invalid.
- CD_GENERATION : generates meaning representation code.

Single statement dialogue is described in the algorithm of Table V and flowchart of Figure 6.5 and includes the following:

- LEX_ANALYSIS : performs lexical analysis.
- SYN/SEM_ANALYSIS : performs syntactic and semantic analysis.

TABLE IV

Menu-driven Dialogue Algorithm

```
ALGORITHM MENU_DRIVEN_DIALOGUE

Begin
write "ENTER SENTENCE?"
if answer = yes then
begin
  do MDD
  else
  do E_INTERRUPT
end if
End
```

* *

```
MDD

while not_end_of_sentence do
begin
  write "ENTER OPTION"
  read option
  if option = legal then
  begin
    do UPD_STATUS
    do SYN/SEM ANALYSIS
    else
    do I_INTERRUPT
  end if
end while
```

* * * *

- SN/SM_INTERRUPT : interrupts when syntactic and/or semantic analysis is incomplete.
- L_INTERRUPT : interrupts when lexical analysis is incomplete.
- U_INTERRUPT : interrupts after user's desire.
- CD_GENERATION : generates meaning representation code.

Complex statement dialogue is described by the algorithm in Table VI and flowchart of Figure 6.6 which include the following:

- LEX_ANALYSIS : performs lexical analysis.

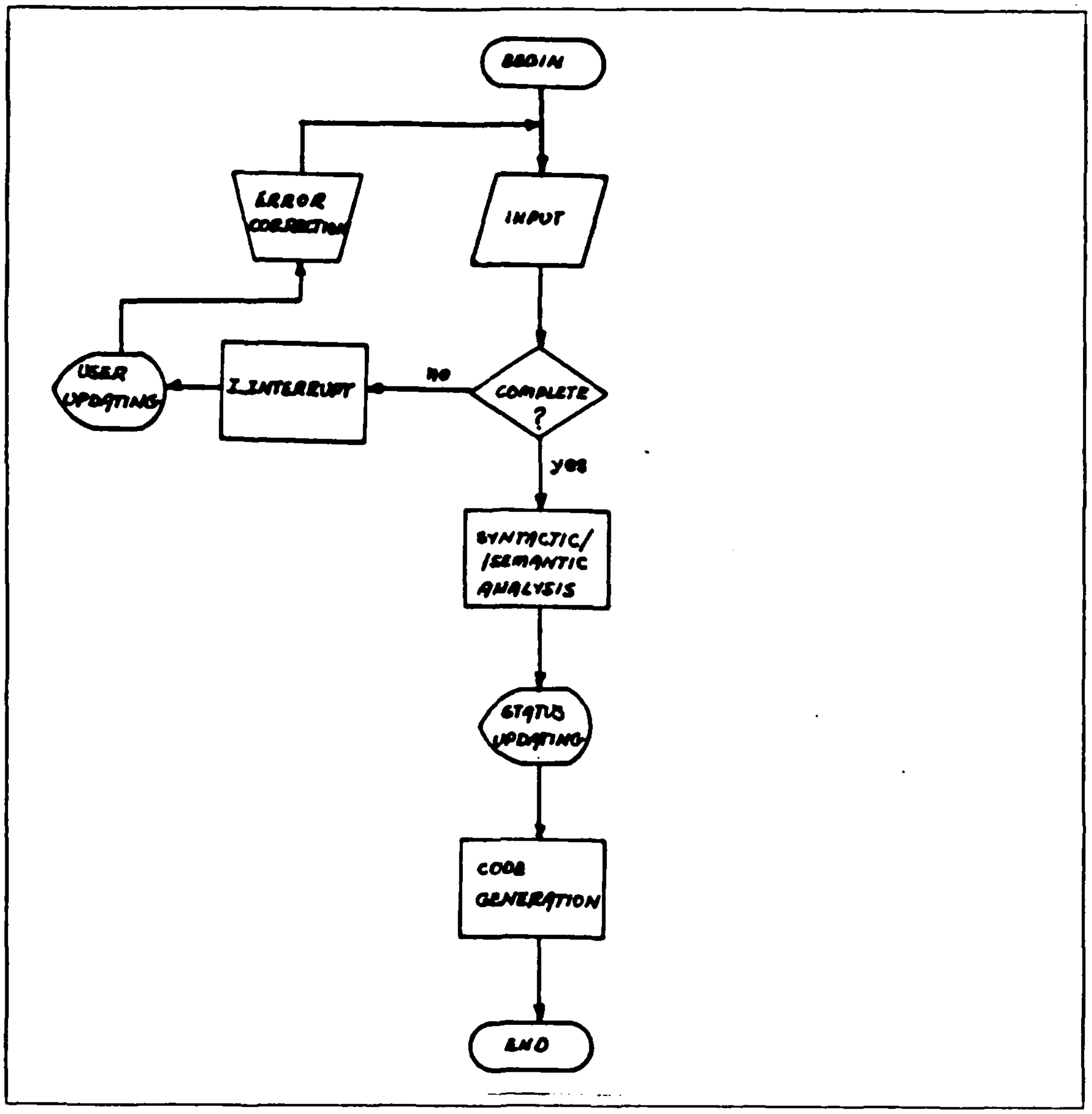


Figure 6.4 Menu-driven Dialogue Flowchart.

- SYN_ANALYSIS : performs syntactic analysis.
- SEM_ANALYSIS : performs semantic analysis.
- DIS_ANALYSIS : performs discourse analysis.
- UPD_CONTEXT : updates context information.
- L_INTERRUPT : interrupts when lexical analysis is incomplete.

TABLE V

Single Statement Dialogue Algorithm

```
ALGORITHM SINGLE_STATEMENT_DIALOGUE

Begin
write "ENTER SENTENCE ?"
read answer
if answer = yes then
begin
  do SSD
  else
  do U_INTERRUPT
end if
End
```

* *

```
SSD

begin
  do LEX_ANALYSIS
  if LEX_ANALYSIS = complete
  begin
    do SYN/SEM_ANALYSIS
    if SYN/SEM_ANALYSIS = complete then
    begin
      write "PROCESSING COMPLETE"
      do CD_GENERATE
      else
      do SN/SM_INTERRUPT
    end if
  end if
  else
  do L_INTERRUPT
end
```

* * * *

- SN_INTERRUPT : interrupts when syntactic analysis is incomplete.
- SN_INTERRUPT : interrupts when semantic analysis is incomplete.
- D_INTERRUPT : interrupts when discourse analysis is incomplete.
- U_INTERRUPT : interrupts after user's desire.
- CD_GENERATION : generates meaning representation code.

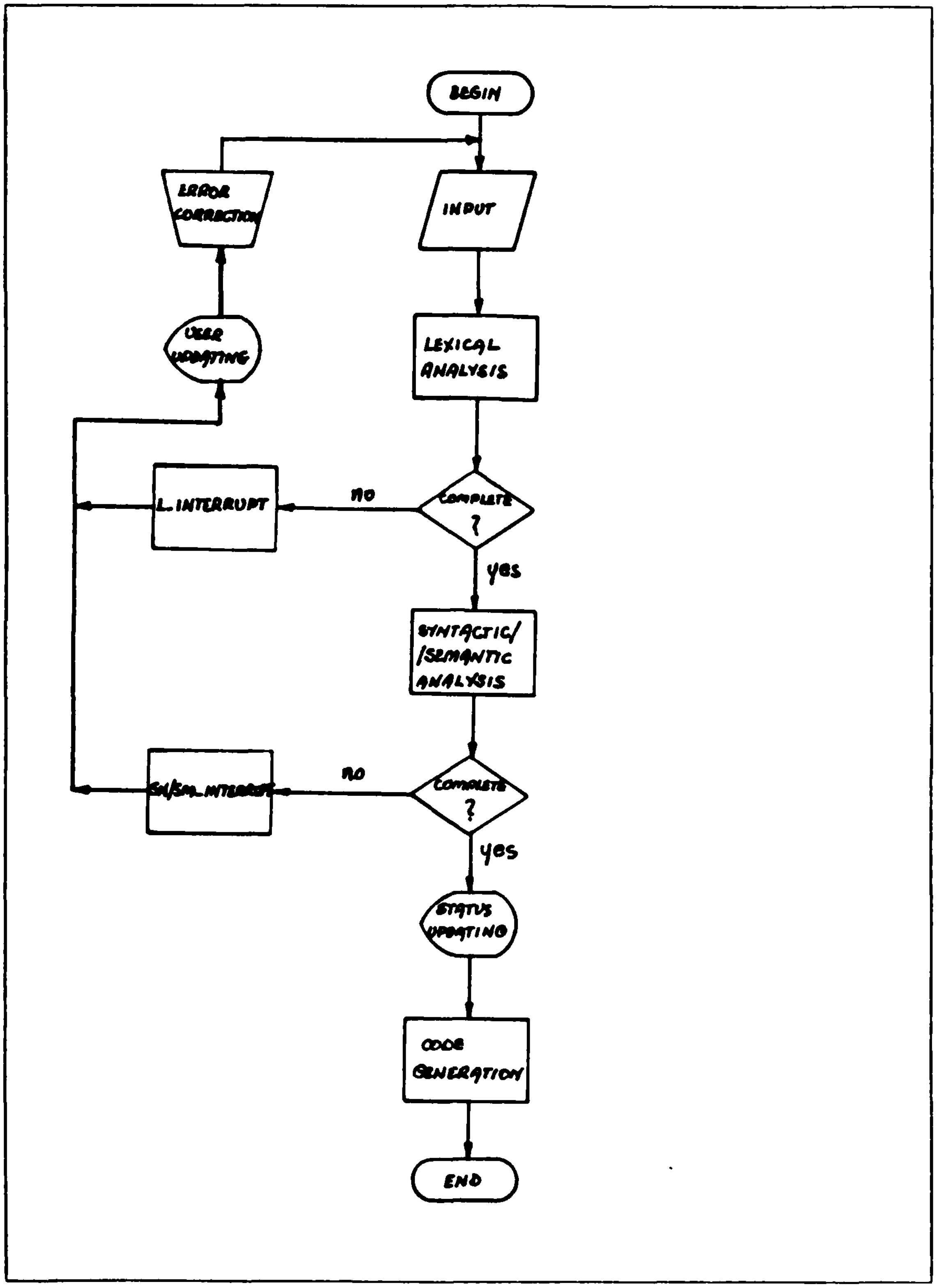


Figure 6.5 Single Statement Dialogue Flowchart.

TABLE VI

Complex Statement Dialogue Algorithm

```
ALGORITHM COMPLEX_STATEMENT_DIALOGUE

Begin
write "ENTER STATEMENT ?"
read answer
if answer = yes then
begin
  do CSD
  else
  do U_INTERRUPT
end if
End
                        * *

CSD

begin
do LEX_ANALYSIS
if LEX_ANALYSIS = complete then
begin
  do SYN_ANALYSIS
  if SYN_ANALYSIS = complete then
  begin
    do SEM_ANALYSIS
    if SEM_ANALYSIS = complete then
    begin
      do DIS_ANALYSIS
      if DIS_ANALYSIS = complete then
      begin
        write "PROCESSING COMPLET"
        do UPD_CONTEXT
        do GENERATE
        else
        do D_INTERRUPT
      end if
      else
      do SM_INTERRUPT
    end if
    else
    do SN_INTERRUPT
  end if
  else
  do L_INTERRUPT
end if
end
                        * * * *
```

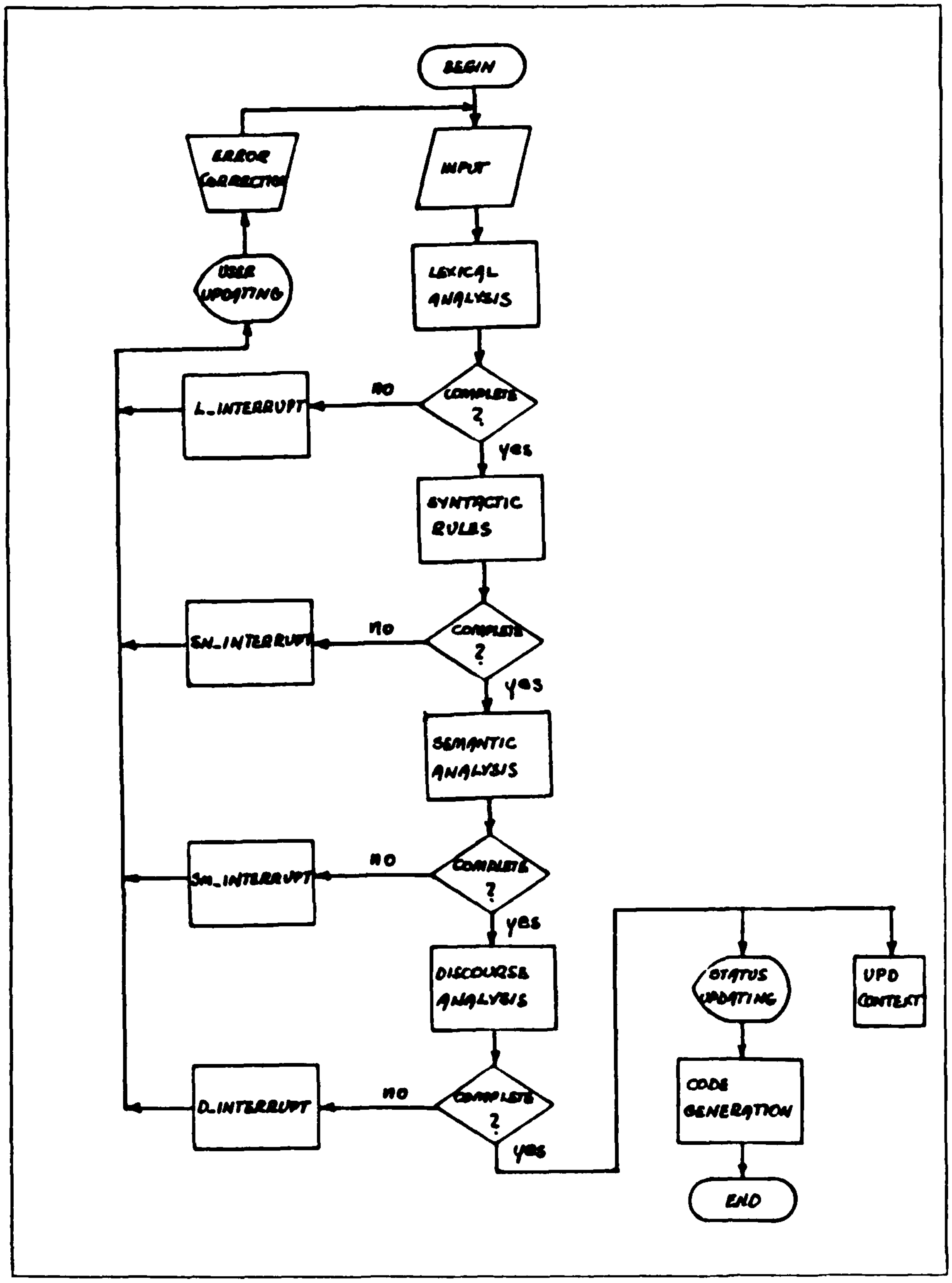


Figure 6.6   Complex Statement Dialogue Flowchart.

VII. THE PRACTICALITY OF NATURAL LANGUAGES

The purpose of this chapter is to examine the practicality of natural languages. We associate the term "practical" with anything that serves a purpose. In this context, the practicality of natural languages can be defined as the degree to which they serve the purpose of effective man-machine communication.

The practicality of natural languages can be examined in four application areas:

- Database querying.
- Natural language programming.
- Natural language dialogues.
- Text knowledge systems.

The idea of the approach is to identify the needs of the users in each application area and then to examine how well the natural languages serve these needs.

A. DATABASE QUERYING

The method which best serves the user is the menu-driven dialogue. The reason is , that although this method is somehow restrictive for the user, it does provide the big advantage that input statements are always within the semantic boundaries of the system. This approach addresses very well the needs of the user for meaningful feedback, minimum mental load and minimum stress for errors. The same method becomes too cumbersome if the natural language subset is large.

Another issue in data retrieval applications concerns answers to queries. In this area the natural language greatly contributes in the presentation of the output in a form which satisfies user's needs for readability and understandability. A requirement which very often arises and which, if it is not satisfied, may lead to erroneous answers, is that the system must take into account the intentions of the user when he/she does the question and respond appropriately by adapting its answer to user's intention. For example, in a question like

Is there any record with Beethoven's 9th symphony?

the system may respond

Yes

In this case it is possible that the customer will ask about the place where he/she expects to find the record. The system could in this case respond like

Yes, shelf #5 in row A

predicting that in a case where the record was available the customer would have asked about the place where the record is.

Another issue in taking answers in natural language is the amount of redundancy built into the answer. There are cases where too much information expressed in natural language hides the important points of the answer and makes the substance difficult to be distinguished.

B. NATURAL LANGUAGE PROGRAMMING

The difference between natural language programming and natural language querying systems is that the first has a broader scope; it has the task of translating indefinitely large subsets of natural language into full-scale programming languages and it is very difficult to limit the English subset.

The role of the programming languages is to provide a vehicle for the user to specify his/her instructions to the computer, in order for the computer to perform a given task. These instructions represent the user's intenstons . Of course natural language which every user knows well, is very good at this point.

The use of natural language as programming tool implies that the compiler will have to carry the load for recognizing the intenstons of the user expressed through the set of the instructions. This load becomes even heavier when user specifies his intentions indirectly. Natural language for example makes use of pronouns, noun phrases and elliptical references very often and any language subset which would not include such parts in its structure would not be too much practical in recognizing user's intentions.

Several experimental systems have been reported. They allow a user to write programs in a <u>restricted</u> subset of English language. A recent example, [Ref. 11], showed that sophomore level programming students were able to program array-manipulation problems faster and with fewer mistakes in English than in PL/C.

A person who writes programs in subsets of English language, feels sufficient freedom for expressing his/her instructions. The problem is that the same difficulty which makes recognizing user's intentions so difficult is the reason which makes providing feedback difficult. This point must become the main focus of the research if it is desirable for the natural language to survive as a programming language.

There are two areas where the use of natural language seems practical. The first concerns the text knowledge systems and it is in some way illustrated by THEIRESIAS [Ref. 12:pp. 87], a computer based consultation system intended to supply near-expert level advice on difficult cognitive problems. The second concerns the use of a grammar to map procedures into English descriptions as an aid to documentation. Natural language gives the capability for bidirectional translation (source-to-target and target-to-source) through the use of a symmetric grammar [Ref. 13].

C. NATURAL LANGUAGE DIALOGUES

Most systems in this area are in their infancy. The reason is that when people communicate between them, they leave many intentions implicit. In any case of human dialogue, theories of speech, acts and intentions come into play and and these theories are the focus of most experiments concerning natural dialogue systems. All these theories have to deal with problems concerned about what is intended rather than what is stated. This implies that some means which allow for tracking user's attention during the dialogue, have to be used. The trouble with this case is that user knows what topic has been chosen but the system must guess from the next sentence, what the topic of the preceeding sentence was.

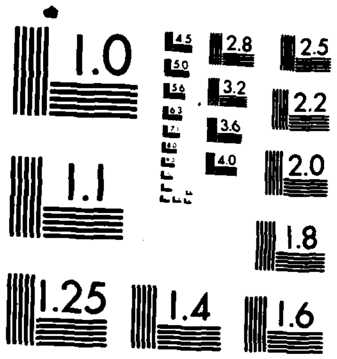
D. TEXT KNOWLEDGE SYSTEMS

This area includes applications which concern automatic translation of natural language text into knowledge data-bases. The primary issue in this area is the same; to recognize the intentions of text's authors.

Texts usually include dialogues, narrations and descriptions. They are written by each author for a certain purpose. Each text has built in the writing style of its author. Good writing styles are in general characterized by minimum redundancy, co-references and elliptic descriptions of events.

The intentions of dialogue participants and authors of narrations and descriptions are not usually reported in the text; they are left to be inferred by the reader. This is exactly the reason why the use of natural language for the creation of text knowledge systems is rather impractical at the present.

Recognizing the intentions of the authors and dialogue participants requires the use of discourse analysis theories and computational techniques which are still in the experimental stage and they must still be further developed before the application of natural language for the production of text knowledge systems become practical. When this moment arrives the user is expected to have access to any portion of the text, answers to questions and summaries of various text segments.


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

VIII. APPLICATIONS OF NATURAL LANGUAGES IN THE MILITARY ENVIRONMENT

In the early 1970's the United States Department of Defense noticed a trend in rising software costs in major systems. The greater percentage , about 56%, of these costs were associated with the embedded computer systems while the other data processing activities took only 19% [Ref. 14].

There were several language-related reasons which contributed to this problem:

- Use of languages which did not adequately support the existing applications.
- Use of languages which did not support modern programming methodologies.
- Use of languages which failed as software development tools.
- Use of languages which failed in dealing with the complexity of the required solutions.

Because the majority of software costs in DoD were associated with embedded computer systems, the DoD turned its attention to that particular area. Embedded computer systems are minor systems within larger computer systems which are not primarily computational, such as weapon and control systems.

Embedded applications have some common characteristics:

- They are usually large.
- They are highly reliable.
- They are physically constrained in relation to hardware address space and speed.
- They have particular programming requirements for parallel processing, real-time control and exception handling.

- Highly experienced personnel.

Although the technology had been improved to a point where very fast hardware and efficient compilers had been developed and promised efficient translation of programming languages, this had not made possible the use of the existing programming languages in embedded systems. At this point, the High Order Language Working Group was established which was assigned the responsibility of determining the requirements for DoD's languages. Conducting a review throughout the military departments, federal agencies and other groups they came out with the result that most groups desired a language which should satisfy requirements like:

- High precision
- Pinpoint accuracy
- Modern programming methodologies support.
- Exception handling support
- Very high reliability.

These requirements led the DoD to sponsor the development of the ADA programming and design language. This language was assigned the responsibility to express programming and design solutions .

Another problem, which is common within the data processing area, appeared in 1980 and is expected to affect DoD's data processing activities, concerns the availability of experienced personnel. This problem is faced by almost every organization today private, public, profit or non-profit. This problem is expected to become worse in the next decade. Some reasons for this shortage in data processing personnel have been identified [Ref. 15] as follows:

- Reduced availability of professional data processing education in the USA.
- Demand for users far exceeds existing availability.
- High personnel turnover.

As an organization with a high degree of data processing activities, DoD is expected to "suffer" in both the embedded systems applications area and the transaction processing, the second biggest applications area. Two questions which arise at this point are:

- To what degree can user-friendly languages, and in particular natural language, contribute to solving the two prementioned problems which DoD faces?
- Does the natural language merit any attempt of implementation within DoD?

Comparing embedded systems characteristics and requirements to the capabilities of natural languages as they have been developed to this point we can easily see that :

- Natural languages do very poor in fulfilling embedded systems requirements, if they do at all.
- They are of trivial importance as means of eliminating the problem of personnel availability. The reason is that embedded systems require highly experienced personnel, which has no need of using natural language and even more it will probably try to avoid it .

These reasons are considered adequate to show that <u>the use of natural language in embedded applications environment is not recommended at all</u>.

The other application area within the DoD concerns transaction processing. These applications are not operated or developed in such a strict and formal environment as embedded systems are. They have some common characteristics, including:

- Fairly general specifications.
- Short response time.
- Accuracy.
- Moderate complexity.
- Personnel of moderate experience.
- Precision.

Looking at these characteristics we see that there are some possibilities for implementing natural language applications but they are very limited. This is a result of the fact that the factors of response time, accuracy and precision are still crucial and natural language applications cannot satisfy these requirements. With the progress performed so far in natural language applications, some hope exists for using natural languages for data retrieval applications. This is justified by the fact that there always will be possibility for having personnel who, at the beginning of their careers will desire a natural language dialogue with the system.

There is, though, an area where natural language is expected to solve some problems. This area concerns training of people who have no data processing experience but they must be used in various positions in computer systems. These people must be in some way educated in handling the computer. Every novice user in front of the computer feels quite uncomfortable and frustrated. He/she feels that he/she is facing a system which is unfriendly and that he/she will probably never be able to dominate it. He/she also feels that he/she has to use the system to perform a task under certain constraints and the system does not care if he/she has any problem in giving instructions to it.

Natural language communication is expected to greately contribute to eliminating such problems. Starting with simple queries in natural language for data retrieval, users will be able to overcome the frustration and stress they are feeling in front of the computer and gradually pass on to dialogues with limited natural language input, ending up after some time with the use of languages like SQL or dBASEII.

As far as programmers are concerned, natural language is not indicated at all, even for educational purposes.

Natural language is still far from the point of usability a programming language. So it is better for these people to start their data processing education with one of the existing programming languages which best fits to the requirements of their upcoming jobs.

IX. CONCLUSION

This research has focused in issues related to the design of Man-Machine Interfaces and particularly, it concentrated on Natural Language Interfaces.

The design objectives have been identified as follows:

- Ease of learning
- Ease of use
- Flexibility with relation to user and task characteristics
- Transparency
- Reliability

As a possible solution for most of the problems in man-machine interaction, the use of natural language is seen as the means for communication of user ideas to the computer. Although natural language had been proved excellent for communication among humans, this does not happen in most cases of communication between human and compuer. The only application area where significant progress has been accomplished, is the data retrieval applications. In the other areas, like Natural Language Programming, Natural Language Dialogue and Text Knowledged Systems, some progress has been made but, almost everything is in the experimental stage.

Based on the general objectives of the man-machine interfaces and approaches to the combination of the components of a natural language understanding system, a general purpose natural language interface has been proposed. This interface is expected to be suitable for all three types of natural language communication tasks:

- Simple statement understanding
- Simple interrelated statements understanding
- Text and dialogue understanding

In order for the interface to address the objectives of man-machine communication and, even more, to satisfy the requirement for transportability, it was found appropriate that the natural language understanding portion will be separate from the application specific part.

It is recommended that emphasis should be given in two points:

- Improving the understanding capabilities of the natural language portion of the system.
- Creating translators which will effectively translate the meaning representation code according to the requirements of each particular application.

Military organizations have to some degree similar processing needs as any other organization and they should be expected to suffer from the lack of competent data processing personnel during the upcoming decades. The contribution of natural language applications to the problem of having people with no data processing experience working at various data processing jobs is not expected to be significant in the immediate future. The only practical application within the military environment and with the progress performed so far in natural language understanding systems, is training of novice personnel in the begining of their careers and switching them to whichever language will be found appropriate when these people acquire some experience in handling the computer.

APPENDIX A
AN OVERVIEW OF THE LADDER SYSTEM

LADDER (Language Access to Distributed Data with Error Recovery) is a computer system which provides answers to queries entered through terminals in a subset of natural language concerning a database with information about ships. The system accepts questions about the data expressed in natural language, and creates sequences of queries to the database management system (DBMS), determines which machines will process the queries, links those machines over the ARPANET, controls processing and prepares answers to the questions.

The more important characteristics of the LADDER, are shown in the Table VII .

LADDER is currently installed on three hosts on the ARPANET; SRI-KL at Stanford Research Institute and ISIB and ISID at the Informations Science Institute at the University of Southern California. It has been operational since June 1976 and undergoes continous improvement.

A demonstration of interaction between the user and the LADDER, is shown in Figure A.1 .

TABLE VII

Important Characteristics of the LADDER

1. Dialogue is controllable by either the user or the system.
2. Input in any combination of upper and lower case letters is acceptable.
3. Ships can be specified by :
   a. Name
   b. Class
   c. Type
   d. Classification
4. Questions may include simple and/or complex restrictions.
5. User is allowed to extend the grammar dynamicaly and handle new types of questions.
6. User is allowed to modify question format.
7. User is allowed to replace a sequence of old questions with a single new question.
8. Segments of sentences can be interpreted in the context of the previous question(s).
9. User can control the amount of output.
10. User can control the "amount" of interaction by changing the value of the verbosity parameter.

```
LADDER: Do you want instructions?
USER  : no

USER  : use dbms20
LADDER: PARSED!

USER  : set verbosity to be -2
LADDER: PARSED!

USER  : give the destination of Kennedy and Whale
LADDER: PARSED!
        Give  the  destination of  ship S   where
        either the name of the ship S is  Kennedy
            or the name of the ship S is WHALE

        DEST = NORFOLK, NAPLES

USER  : how long will it take  for the  Wainwrite
        to arrive in Norfolk?
LADDER: ...
        Trying Ellipsis: Ellipsis has failed.
        LADDER does not  expect the  word "TO" to
        follow "HOW LONG WILL IT TAKE"
        Options for this word or symbol are:
        <SHIP>, FOR

USER  : how long will it  take for the  Wainwrite
        to arrive in Norfolk?
LADDER: PARSED!
        Give the location of the port NORFOLK
        Give the normal  cruising  speed  for,
                 maximum cruising speed for,
                 position and the time at position
                for the ship WAINWRITE.

        from (NOSPD 16 #HRS 276  MXSPD 34 HRS 130
             POSITION 3700N01700E DATE 17 Jan 79
             1200)

USER  : done
LADDER: EXIT
```

Figure A.1   Demonstration of Interaction with LADDER.

LIST OF REFERENCES

1. Weinberg G., The Psychology of Computer Programming Van Nostrand, 1971.

2. Pressman S. R., Software Engineering A Practitioner's Approach, McGraw-Hill Book Company, 1982

3. Jarke M., Krause J., Vassiliou Y., Studies in the Evaluation of a Domain Independent Natural Language Query System, New York University, 1984

4. Martin J., Design of Man-Computer Dialogues, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1973

5. Dehning W., Essig H., Maass S., The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs, Spring-Verlag, Berlin, Heidelberg, New York, 1981

6. Tanner W., User's Guide to TOPS-20, USC Informations Science Institute, 1982

7. Rich E., "Natural Language Interfaces", IEEE Computer, Vol. 9, 1984

8. Woods W. A., "Transition Network Grammars for Natural Language Analysis, Communications of the ACM, Vol. 13. 1970, pp. 591-606

9. Sager N., Natural Language Information Processing, Addison-Wesley, Reading, Mass., 1981

10. Lehnert W. G., Ringle M. H., Strategies for Natural Language Processing, Lawrence -Erlbawm Associates. New Jersey

11. Biermen A. W., Balzard W. B., "Toward Natural Language Computation", American Journal of Computer Linguistics, Vol.6, pp. 71-86, 1980

12. Sime M. E., Coombs M. J., Designing for Human-Computer Communication, Academic Press, 1983

13. Simmons R. F., Computations from the English, Prentice-Hall Inc., New York, 1984

14. Booch G., Software Engineering with ADA, The Benjamin/Cummings Publishing Company Inc, 1983

15. Cash I. J. Jr, McFarlan F. W., McKenney L. J., Corporate Information Systems Management, Richard Irwin Inc., Homewood, Illinois, 1983

BIBLIOGRAPHY

Bolc L., Natural Language Communication with Computers, Lecture Notes in Computer Science, Spring-Verlag, Warsaw, 1978

Fairley E. R., Software Engineering Concepts, McGraw-Hill Book Company

Garvin L. R., Natural Language and the Computer, McGraw-Hill Book Company, 1963

Meadow C. T., Man-Machine Communication, Wiley-Interscience, 1970.

Norman A. D., Design Rules Based on Analysis of Human Error, Communications of the ACM, Vol.26 No4, April 1983

Senn A. J., Information Systems in Management, Wadsworth Publishing Company, Belmont, California, 1982

Wasserman I. A., Some Principles for User Software Engineering for Information Systems, IEEE Digest of Papers, 1975

Wetherbe C. J., Systems Analysis and Design, West Publishing Company, St. Paul, Minnesota,1984

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Computer Technology Curricular Office Naval Postgraduate School Code 37 Monterey, California 93943-5100	1
4.	Chairman, Code 54 Gk Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
5.	Associate Professor Tung X. Bui, Code 54 Bw Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
6.	LCDR Barry A. Frew, Code 54 Fw Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
7.	Embassy of Greece Naval Attache 2228 Massachusetts Ave., N. W. Washington , D. C. 20008	7
8.	LT Ioannis Kotrozos 26 Marinou Antypa Str. Neon Iraklion Attikis Greece	1