USING RESTRICTED NATURAL LANGUAGE FOR DATA RETRIEVAL: A PLAN FOR FIELD EVALUATION

Jon A. Turner, Matthias Jarke, Edward A. Stohr, Yannis Vassiliou, and Norman White

May 1982

Center for Research on Information Systems
Computer Applications and Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #38

GBA #82-51(CR)

Presented at NYU Symposium on User Interfaces, New York University, Graduate School of Business Administration, New York City, May 1982.

Using Restricted Natural Language for Data Retrieval:

A Plan for Field Evaluation

Abstract

One strategy that has been proposed for dealing with the growing backlog for development of applications is to give casual users languages for interacting directly with databases. Yet, there is little agreement on the form such languages should take. Should they be natural-like, conforming closely to a user's native tongue or should they be structured to take advantage of the characteristics of formal languages?

This paper presents the rationale for and design of a field evaluation of natural language for data retrieval. The natural language system and application are described along with the research design of the project. The results of the first part of the study, a laboratory experiment to investigate whether users perform better with an artificial or natural language, suggest that after equal amounts of training no difference in subject performance is found between languages using a paper and pencil test. The insights gained to date are summarized.

Acknowledgements

This work was carried out as part of a joint study being conducted with the IBM Information Systems Group in White Plains, N.Y. The assistance of John Kesick and Ken Michielsen is appreciated. We are indebted to our principal users, Ann Dauberman, Associate Director of External Affairs, and Jay Avelino, Director of the Graduate School of Business Administration Alumni Association, for their support and time, and to Joe Ciciollo of the Administrative Data Processing Center for his assistance.

1.0 INTRODUCTION

A major concern of both managers and researchers in Information Systems is productivity. While the cost of a unit of hardware performance has decreased by a factor of more than 100 in the last ten years, the cost to develop application systems has remained about constant. Structured methodologies for analysis and programming; building techniques such as prototyping, database management, and programmer work benches; and new methods for organizing development work have not substantially reduced the time or cost to develop application systems [1]. The inability to improve development productivity has contributed to a backlog of applications for implementation. By one estimate, this backlog will take 3-4 years to remove without adding any new requests [Alloway and Quillard 1982].

One appealing strategy for dealing with a portion of this implementation backlog is to encourage end users to directly access their own data. This approach would reduce the need to write report programs and permit users more timely access to data. Yet, there is little agreement on what form a computer interface for casual users of these databases should take. The extensive training required in current higher level languages, such as COBOL or PL/1, acts as a barrier to use. It is hard to imagine large numbers of middle level executives being trained in these languages and using these skills effectively to access data. Special database query languages, such as SQL or Query By Example, while being easier to learn and more powerful for data retrieval than either COBOL or PL/1, still require considerable training and practice to be used effectively.

An approach to this problem is to create a 'natural' (that is, English like) computer interface language which would permit users to perform their tasks directly. Because 'novice' users who are applications specialists presumably know English, no or little training would be required in either the interaction language or the application domain, since the specialist knows these already.

Critiques of this approach observe that English, as commonly used, lacks the precision needed in an interface language and that the meanings of English statements are frequently ambiguous. Yet, a system purporting to permit English language data access has been commercially available and is reported to have over 40 paying customers [Computerworld 1981]. The question, then, is not whether such systems are feasible, but how well they work, and how to make them work better.

In the following sections we present arguments for and against natural language for data access, the description of a research project at NYU intended to explore the use of one natural language system in a real application setting, and preliminary findings.

2.0 NATURAL LANGUAGE FOR DATA ACCESS

The assumption underlying much of the argument for natural language as a computer system interface appears to be that it will make a system accessible to a 'naive' user without the necessity of specialized training [Malhotra and Wladawsky 1975]. Because of an unfamiliarity with computer systems and a lack of time, opportunity or

motivation to learn a special interface language, a 'naive' user is unable to take advantage of computer systems to aid in task accomplishment. Natural language is presumed to remove this barrier.

For some computer applications, natural language appears to be a good form of communication. For instance, Petrick (1976) analyzed information retrieval requests of a large company. Questions were submitted in writing to clerks who translated them to formal query language equivalents, submitted them to a computer system, and returned the resulting answers to requesters. Requests were of the form:

> Who is the buyer on PO H2394?
> What is the total amount of dollars outstanding on POs for supplier 20035?

Petrick concludes that if such requests could have been processed in their original form, answers would have been returned to requesters sooner with less human effort. What is surprising is that the requests appear to have similar structures suggesting that once these patterns were recognized, the problem of interpretation could be simplified.

2.1 Arguments Against Natural Language

The arguments against natural language as a computer interface center on the observation that English, as used in practice, frequently has multiple or ambiguous meaning; uses fuzzy words, such as 'similar' or 'almost'; and permits partial specification with the listener filling in the missing information based on his understanding of the context of the statement [Malhotra and Wladawsky 1975].

Another point made against natural language is that it will encourage users to make requests beyond the language, data, or knowledge boundaries of the system [Shneiderman 1981]. Granting that unrestricted natural language is technically unfeasible in the foreseeable future, subsets of natural language can be successfully used for communicating with computers [Petrick 1976]. Users, because of imperfect knowledge about the coverage of a system, will make references outside the system domain, but that will merely result in no answer, a clarification request, or an incorrect answer. Restrictions, then will have to be learned, defeating one of the advantages of natural language. This may lead to proactive interference, the confusion between what persons know and what they are trying to learn [Shneiderman 1980].

The third argument raised is that the structure of an artificial language may aid in problem formulation [Shneiderman 1981]. However, large differences in problem solving styles among individuals [Miller 1981] suggest that some users may pay a price in translating their solutions to the structural form of an artificial language.

2.2 Discussion

We have several observations on the foregoing debate. First, the purpose of a system (the task(s) for which it is to be used) is a major factor in determining whether natural language is likely to be a good interface. We suspect that for procedural programming, natural language is too ambiguous and verbose to be effective, as suggested by studies of procedure manuals or recipes [Miller 1981].

One category of systems where natural language may be useful is Question-Answering systems. Simmons (1970) describes them as consisting of 1) accepting natural language statements as input, 2) transforming the statement into formal language by syntactic and semantic analysis, 3) providing deductive or inductive reasoning procedures for such operations as answering questions, and 4) generating an English string as an answer.

Question-Answering systems are a more appealing category than procedural programming because the desired output product can frequently be concisely and unambiguously specified. Questions tend to follow prescribed forms that define outputs and often refer to bounded domains, sometimes defined by the contents of a database. Describing the procedure for producing output involves conveying more information, and different categories of information, such as sequence and control, instead of just content. Therefore, much of the concern about the adequacy of natural language as an interface language recedes when the task involves goal rather than procedure description.

Another factor that influences whether natural language is a good interface is the user's skill level. Shneiderman (1981) distinguishes between two types of knowledge; syntactic and semantic. Syntactic knowledge refers to a user's skill with a particular interface language. Semantic knowledge is the user's familarity with the specifics of an application domain. Shneiderman speculates that users with little knowledge of an interface language (low syntactic knowledge) and a lot of application domain knowledge (high semantic knowledge) are the best candidates for natural language. Malhotra and

Wladawsky (1975), on the other hand, suggest that natural language should be most useful in the low syntactic knowledge, low semantic knowledge case, because this makes the least demand on a person for specialized knowledge. It is our opinion that with Question-Answering systems some syntactic knowledge, either about the form of the interface languages or its restrictions, and semantic knowledge of application domains will be needed to construct meaningful questions in Natural Language.

2.3 Research Summary

Much of the research about computer languages that pertains to the issue being investigated in this study can be divided into two categories: research about language skill acquisition (learning) and research about using language in task performance.

Skill Acquisition Studies. Reisner (1977) compared the 'learnability' of two formal languages, SEQUEL [Astrahan and Chamberlin 1975] (currently called SQL) and SQUARE, differing in syntax (e.g., SEQUEL uses English key words while SQUARE uses positional notation). After training, a battery of tests were given to 61 students. Reisner concluded that both programmers and non-programmers could learn to write queries in either language; non-programmers performed better in SEQUEL than in SQUARE (programmers scored higher than non-programmers for both languages combined). She also concluded that the features of languages differed considerably in learnability and recommended that SEQUEL be treated as a layered

language with the easier layer intended for users of limited sophistication or need.

Welty and Stemple (1981) examined the relationship between a language's procedurality and programmers performance with the language. Subjects were trained in two language: TABLET, a procedural language that requires specifying operations to be performed on a base relation; and SEQUEL, a less procedural language based on key words. The procedural language subjects (TABLET) had significantly higher average scores on difficult queries than did the less procedural language subjects (SEQUEL) on both a test at the end of the training period and on a retention test. There was no significant difference in performance of the subjects on easy queries. Differences in performance were also found on a language feature basis. That is, performance differences depended on the language features used by subjects to answer queries. This is consistent with Reisner's (1977) finding that the features of languages differed considerably in ease of learning. They also found that the more procedural language was easier to learn for subjects with no previous computer language exposure and that procedural language subjects with experience showed less of a drop-off when tested for retention than did less procedural language subjects, also with experience. Welty and Stemple concluded that TABLET's procedurality encourages subjects to think in terms of concrete procedures to transform tables of information, which permitted them to perform somewhat better.

Shneiderman (1981) briefly trained subjects in SEQUEL and then tested them in an experiment to determine whether they asked more valid queries in English than in SEQUEL. He found no significant difference in the number of valid queries asked, but did find an order effect with the English-SEQUEL group having more errors than the SEQUEL-English group.

In summary, it appears that subjects can be taught to use a variety of query languages although certain features may be more difficult to teach than others. The procedurality (or structure) of a language seems to be an aid in use and retention. Also, both the skill level of subjects (i.e., the extent of prior programming experience) and the difficulty of queries influences subject performance.

Task Performance Studies. Lehmann et al. (1978) described several field evaluations of Natural Language for database access using the User Specialty Language (USL). Subjects in the first study performed statistical analysis of data on adult education and life. Subjects in the second study were investigating the use of grades in certain courses as predictors of performance in later examinations. The three most frequently used USL functions/grammatical structures in the first study were: 1) verb 'have', 2) apposition, and 3) questions of the form 'how many?'. In the second study they were: 1) prepositional phrases, 2) 'how many?' questions, and 3) verbs other than 'are', 'be', or 'is', i.e. domain specific verbs. [Lehmann et al 1978] concluded that users could adapt to restrictions in a system more easily than expected. It appears that the task being performed

influences the frequency of the grammatical constructs used.

Krause (1979), reporting on a continuation of the second USL field experiment, found an overall error rate of 6.6 percent for one user submitting 2214 queries. He observed that this error rate is much lower than that found in other studies and speculated that error rates may be lower in real applications than in pencil and paper tests. He notes that there is great variation in performance among subjects, with the percentage of errors in one study [Thomas and Gould 1975] ranging from 7 to 77 percent.

Considering the range in performance attributable to individual differences, studies with only one subject are suspect. There was also evidence in the Krause study that the subject followed a 'sufficing' strategy in that once a combination of queries was found that produced the desired result, it was not varied.

Damerau (1980, 1979) described the results of running the Transformational Question Answering System (TQA, formerly REQUEST) in a city government planning department. Of 788 queries posed to the system over a 12 month period, 513 or 65 percent were successfully completed. No information is given on how subjects were trained, what assistance they were given during the experiment or how queries were scored.

In summary, it appears that users can learn to use a query language for task performance, but of all of these studies, only Reisner (1977) and Welty and Stemple (1981) compare the performance of subjects using different languages. Since situational, task, or

individual difference factors potentially exert such a strong influence, and because of the absence of experimental controls, it is not possible to tell from any of these studies how natural language systems for database access are likely to perform.

While these studies provide some useful insights about natural language much remains to be investigated. As Petrick (1976) observes:

> To date there have been few instances of natural language programming, question-answering, or database management systems where efforts were directed toward truly practical application. Most natural language question-answering systems have dealt with toy problems and/or databases for which there exists no body of present or potential computer users who have questions or commands in which they are vitally interested (p. 315) [2].

He goes on to note that there have been few attempts to evaluate the capacity of a natural language question-answering system to satisfy the needs of a user community.

Tennant (1979) is also critical of the lack of exploratory studies:

> The lack of evaluation of natural language processing research leave several critical questions about the work unanswered. Readers are unsure what concepts are included in the system, what accomodations have been made for language variations between users, the restrictions on the discourse domain or database, the restrictions on data manipulation capabilities, and the restrictions on inferencing capabilities. There is usually no information about the match between facilities included in the system and the actual needs of the users. In addition there is little information on what kind of performance would be required of a natural language processor to allow users to carry out tasks at various levels of complexity (p. 3).

The joint program of research being conducted by members of the Computer Applications and Information Systems Area at NYU and the IBM Corporation will provide some further insights concerning the performance of practical Natural Language Question-Answering systems.

3.0 PROJECT DESCRIPTION

We are interested in the following questions. Under what combinations of task and user demographic characteristics, if any, will Natural Language Question-Answering systems work in real world settings? Should Natural Language systems prove practical, then, under what conditions, if any, are they superior to artificial languages? Will the functional capability of a computer interface language change the problem solving behavior of subjects using the language?

As a research approach to investigating these questions we believe that exploratory studies in real work settings offer the most likely means of identifying critical issues for more detailed study in laboratory experiments. We see the combination of exploratory field evaluations paired with laboratory studies as a strong research strategy.

Our approach is to select an application and user population that, based on earlier research, is most likely to benefit from natural language as a computer interface. If a successful system can be built under these conditions then application and user types can be varied to determine how general are the findings.

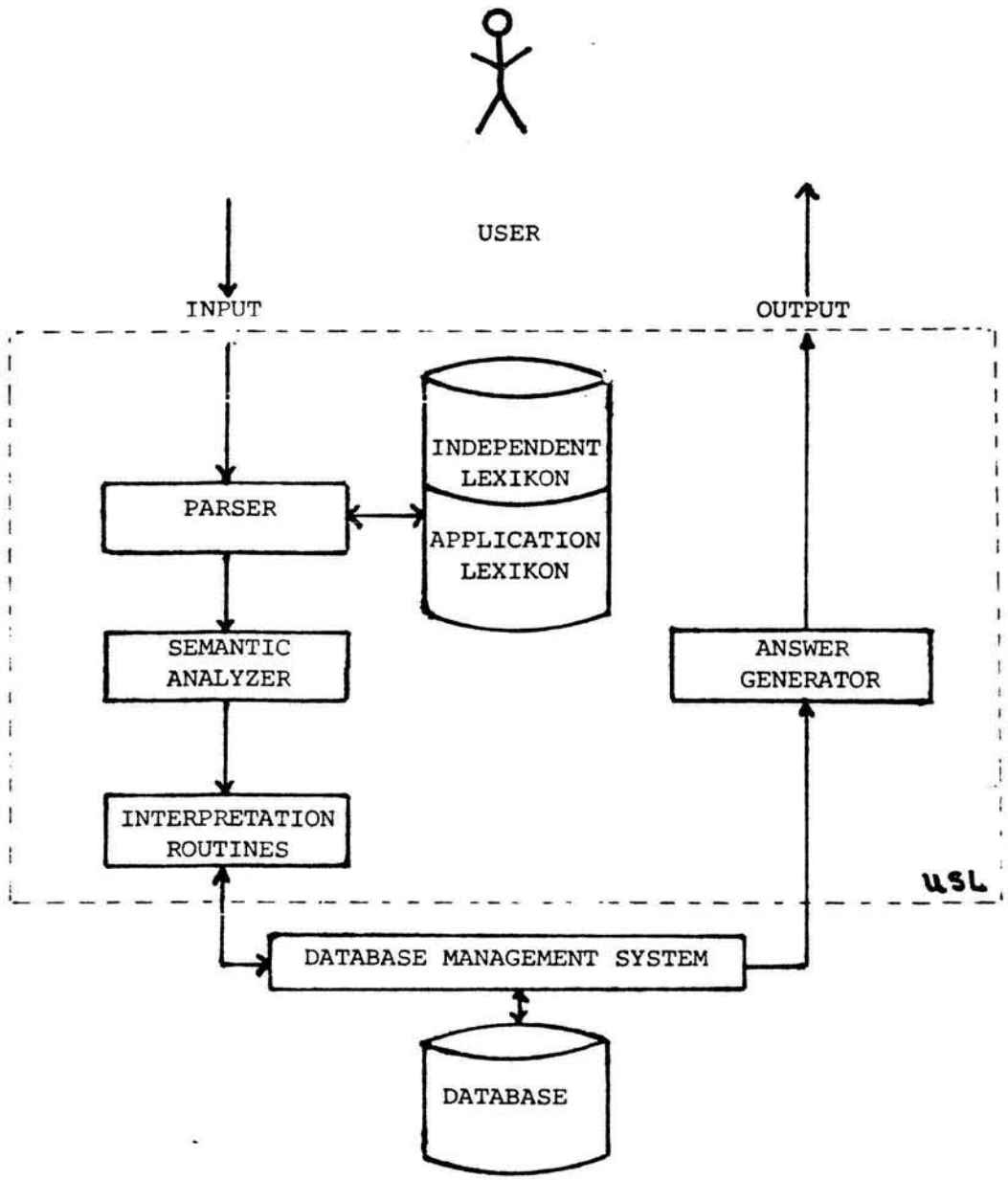

FIGURE 1: Structure of USL

3.1 Natural Language System

The User Specialty Language (USL) developed by the IBM Germany Scientific Center was used as the basis for designing the natural language application system used in this study. USL consists of a revised form of Kay's (1967) bottom-up parser and a function generator that were originally used in the REL system [Thompson and Thompson 1975] and have been extended into a tool for language development under the name of USAGE (User Application Generator; [Bertrand et al 1976]).

An English grammar [3] comprising some 800 rules in BNF specifies both a syntactic configuration to be used as a condition for application of the rule and one or more parameters that replace the original syntactic configuration after the rule has been applied. Each rule references one or several of 70 interpretation (semantic) routines that map linguistic constructs into a formal database interface language (SQL). An underlying, independent relational data base management system accepts these SQL language expressions and returns results which are interpreted in an answer generator.

A lexicon contains all relevant function words whose meanings are independent of particular applications (e.g., prepositions, conjunctions, 'to be', 'to have', days of the week, etc.). Attached to it is an 'application lexicon' containing all those words specific to the application including nouns, verbs, and adjectives. In addition, it includes non-standard plural forms for nouns, non-standard verb tenses, prepositions used with nouns, and other surface structure contextual associations, as well as synonyms (see

Figure 1). Thus, to configure a new application one constructs (or augments) the application specific lexicon and defines the relations for the application in the database system. The fact that these lexicons are separate, greatly reduces the time required to build a new application. Lehmann (1978), and Ott and Zoeppritz (1979) provide more detailed descriptions of USL.

Parsing produces one or more trees whose structure reflects the surface structure of the input sentence. Each node of the tree contains the name of an interpretation routine that is called consecutively, resulting in an intermediate tree structure that no longer only reflects the surface structure, but includes some semantic information. This intermediate tree permits coordination, quantification, and possessive pronouns to be handled more adequately than by operating directly on the parse tree [Lehmann 1978].

Words in the input string may represent names of relations, attributes of relations, or specific values within relation tuples. Common nouns, verbs and adjectives are defined by association with attributes of real or virtual relations in the database. Each attribute of a virtual relation has a defined 'domain' and 'role'. Standard domain types consist of: ZAHL (number), WORT (word, character string), DATUM (date, time of day), and CODE (numeric code). Standard role names include: NOM (nominative case, set of objectives referred to by noun or adjective), ACC (accusative case), VON (genitive attribute), as well as location and time. Prepositions also receive a role name, for example, DAT for indirect objects. Proper nouns and numbers are recognized by default as values in a relation.

Virtual relations (views) are constructed by prefixing the domain and role names for each attribute.

Consider the following example. Suppose we have a base relation:

GIFTSUMMARY (DONOR, AMOUNT, FISCALYEAR)

The verb 'give' can be defined in USL by first establishing a relational view in SQL using the statement:

```
DEFINE VIEW GIVE (WNOM-DONOR, ZACC-AMOUNT, DTP-FISCALYEAR)
AS SELECT DONOR, AMOUNT, FISCALYEAR FROM GIFTSUMMARY;
```

Here the prefix WNOM defines DONOR as a character string (W) in the nominative case (NOM). Similarly AMOUNT is defined in the accusative case with a number domain (Z). Finally FISCALYEAR is defined as denoting a point in time (TP) with domain date (D).

After the process has been completed USL will be able to interpret questions such as: 'Did Smith give 5000?', or 'How much did Smith give in 1981?'

At its current stage of development, USL does have restrictions and limitations. It is frequently difficult to recognize underlying meaning relationships in the scrambled and incomplete forms that natural language queries often take [Petrick 1976]. Phrase-structure systems (i.e., the approach to Natural Language understanding used in USL) must deduce intended meaning from the grammatical structure of expressions, which is extremely difficult. USL does simulate a transformational grammar in its interpretation (semantic) routines for certain conditions, e.g. comparisons.

Another limitation of USL is that only one production rule is associated with each production whereas other approaches permit multiple rules to be passed down as well as up the tree permitting a richer interpretation of meaning. Furthermore, the only way to reference between queries (intersentential) is to create a temporary variable and refer to it in subsequent queries [4].

Besides limited linguistic diagnostic messages, little attempt is made in USL to resolve ambiguous meanings or missing grammatical constructs. The limited user feedback makes it extremely difficult to know the system state and to formulate corrective strategies. Finally, USL has no provision, outside of the analysis performed in the interpretation routines, for drawing inferences.

USL does appear to be less 'ad hoc' then ATN (augmented transition network) based systems (such as INTELLECT [AIC 1982], formerly ROBOT) and USL does contain some general world knowledge in the form of its application independent lexicon or grammar. It also maintains a systematic correspondence between sentential structure and meaning. USL represents a trade-off between ease of application design and limited use of context.

For a more complete description of USL capabilities and limitations see [Stohr et al 1982]. A more detailed comparison among different approaches to natural language systems can be found in [Petrick 1976].

3.2 Application Description

The application selected for development was a Question-Answering system about Alumni of the Graduate School of Business Administration (GBA) at New York University. The system maintains demographic and giving history of school alumni, foundations, other organizations, and individuals. The school has over 40,000 graduates as well as some 5,000 non-graduates who have given to the school over the past 20 years.

Questions about the school's alumni and their giving flow to the Associate Director for External Affairs, located in the Dean's Office complex, originate from faculty, the Deans, student groups or other parties. Either the Associate Director has the information in reports or she calls the school representative at the Alumni Federation, located at the Washington Square Campus some two miles away. If the representative does not have the information he may request a special report from the Administrative Data Center which maintains an Alumni Records System for the University. The request for a special report can take several weeks, since reports tend to be batched together until enough have accumulated to make a complete pass of the master file worthwhile. Periodically, the Center prepares standing reports, and returns them to the Federation, which distributes them to requesters. Gifts are processed centrally at Washington Square and periodically posted to the master file.

Data is extracted from the University system every several months and used to load a Natural Language Question-Answering system.
The actual application contains the following four base relations:

Prospect Master	- name, id, demographic data	- 20,500 tuples
Gift Summary	- id, giving history summary	- 65,000 tuples
Education	- id, education history	- 22,000 tuples
Dictionary	- data element name, description, codes and code meanings	- 1,500 tuples

Figure 2 shows the base relations and the relationships among them as an entity-relationship (ER) diagram. There are approximately 147 virtual relations or 'views' defined (see p 14), the maximum permitted by the underlying prototype database system. The domain of discourse includes alumni and non-alumni who have given to the school, their giving histories, their education, their demographic data (in the indicative record), their role as solicitors, and their role for matching gifts.

3.3 Environment

The application runs in 8 million bytes of virtual memory on an IBM 4341 Group I (4M) under VM/CMS. A number of other applications, including a registration system, run on the same machine. The system is accessed remotely over 300 Baud dial-up lines using printing terminals.

3.4 Research Design

Initially it was thought that Deans and Development Officers would directly use the system. However, it quickly became apparent that principals did not have the time or the patience to participate in a research project. Also, we became concerned that the system would have only two users, the Associate Director of External Affairs

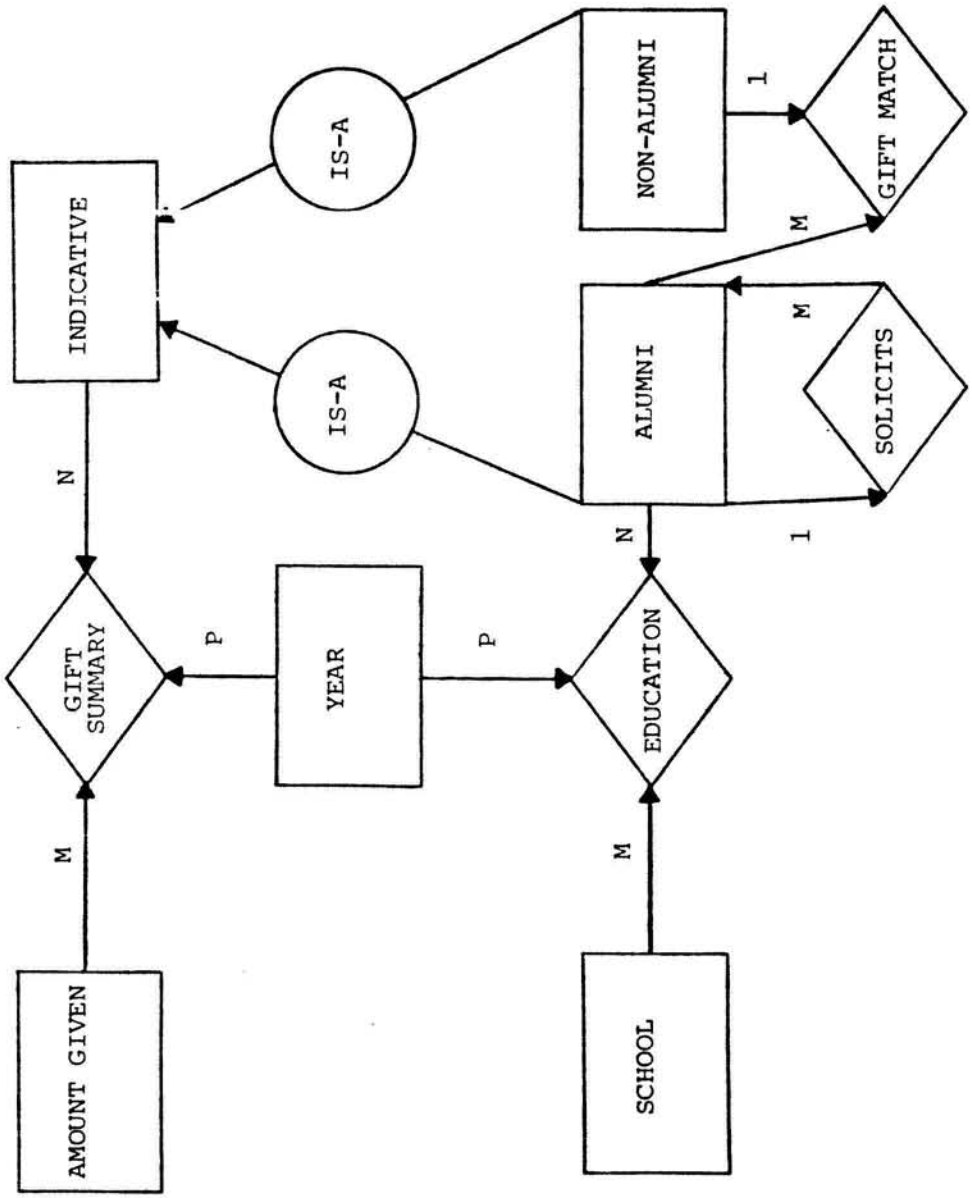

Figure 2: Entity-Relationship Diagram of the Alumni Records Application

and the Director of the GBA Alumni Association, too few for valid results.

Intermediaries. In order to increase the number of subjects and to have better control over data gathering it was decided to use paid subjects to act as intermediaries on behalf of principals. Subjects would meet with principals and obtain a verbal information request. They would then interact with the system to obtain an answer, by typing in one or more queries in the retrieval language, and return the answer to the principal. This approach minimizes the amount of time principals had to devote to the project and isolates them from the instability of a prototype system. If results were positive, principals could directly use the system.

Comparative Study. In field studies the challenging issue is to control for factors not directly measured since it is difficult, especially in exploratory studies, to anticipate what factors will influence outcome variables. Rather than attempt to evaluate a natural language application in the absolute, it was decided to compare the performance of subjects using natural language to the performance of another group of subjects using a reference artificial language, both groups working with the same application. We reasoned that many factors, such as the physical environment, that could affect outcome variables, would tend to effect both groups of subjects equally. If a difference in performance between treatment groups was detected it could more easily be attributed to differences between the interface languages. By selecting a reference artificial language that others had studied it would be possible to validate some of our

results. Since USL mapped natural language queries to SQL for database access, and SQL (or its predecessor, SEQUEL) had been extensively studied (Reisner, 1977; Welty and Stemple, 1981), it was decided to use SQL as the reference (comparison) language.

Treatment Design. Because other researchers had found performance among individuals to be highly variable, a counter-balanced design that would enable us to verify between-group contrasts with within-group contrasts was selected. Figure 3 shows the research design. Subjects were divided into two treatment groups, Group 1 and Group 2. Groups were trained in the application domain then trained in either USL or SQL, tested, and then interacted with principals. At the end of a six week period (hopefully long enough to overcome learning effects), subjects would cross languages. They would then be trained in the new language, tested and again interact with principals. At the end of another measurement period subjects would be given a refresher in the original language and would then interact with principals using which ever language they preferred. During measurement periods subjects would meet with principals twice a week for fifteen minutes each and then have between one to two hours on the system to answer the request.

In order to perform valid comparisons between groups, either a way had to be found to reliably classify requests or a method developed to insure that both treatment groups were attempting to answer the same request. Since classification schemes introduce a degree of uncertainty into the research results a paired design was selected. That is, two subjects, one from each treatment group would

TREATMENT								↓					
GROUP 1	X1	X2	01	X4	02	X3	03	X4	02	X5	02	04	05
GROUP 2	X1	X3	01	X4	02	X2	01	X4	02	X5	02	04	05

X1 - Application Training
X2 - USL Training
X3 - SQL Training
X4 - Serve Clients
X5 - Serve Clients With Either USL or SQL
01 - Pencil and Paper Test (Laboratory Experiment #1)
02 - Measure Performance
03 - Paper and Paper Test
04 - Questionnaire
05 - USL and SQL Retention Test

FIGURE 3

MULTI-FACTORIAL, REPEATED MEASURE, BALANCED DESIGN FOR THE COMPARATIVE EVALUATION OF NATURAL LANGUAGE QUESTION-ANSWERING SYSTEMS

↓- Status of field experiment-May 1982

meet with a principal and be given the same assignment.

It became apparent that the analysis would be performed on two levels: at the query level and at the request level. Queries are the lowest level of interaction with the system and one portion of the analysis would concentrate on describing query level performance in each treatment group. If queries represent the components of work tasks, then requests were jobs to be done. Analysis at the request level would permit identifying how patterns of queries were used in different problem solving situations.

Advertisements were placed at Graduate School of Business and at the College of Business at Washington Square. About 20 candidates were interviewed by members of the research team and eight were selected as subjects for the study. Subjects were selected (for the purposes of control) on the basis of their similarity, except that there are an equal number of women and men. Subjects were given a brief description of study goals and asked to sign the human subject disclosure form. They were paid in two equal amounts for participation in the project.

Subject age varied from 22 to 30 years with a mean of 24.4 years. Subjects had a small amount of prior computing experience; enough to ensure they were generally familar with computing, but not enough to be an expert. The most experienced subject had written 15 BASIC programs and had minor familarity (1 - 4 programs) with another programming language. No one had used more than two hardware systems and none had worked as a professional Systems Analyst or Programmer. Previous work experience ranged from 1 to 7 years with a mean of 3.3

years. Subjects were assigned randomly to treatment groups. We believe the subjects are typical of business or professional people early in their careers, a group that is viewed as one likely to directly use computer technology in their jobs.

In summary, our review of previous research indicated a gap in empirical studies of the use of Natural Language Question-Answering systems in real would settings. Little is known about how well these systems work, what difficulties users and designers encounter, and how these systems compare to other interface alternatives. There is also a need to develop techniques for system evaluation. Finally, an unexplored issue in the literature is the relationship of laboratory experiments to field studies. It would be useful to understand better how the results of laboratory studies could be extended into field settings. The coordinated studies planned for this project should move us toward these goals.

4.0 LABORATORY EXPERIMENT

The primary purpose of the experiment was to determine whether the subjects had achieved a level of language proficiency that was high enough to permit proceeding with the next stage of the field experiment; direct client contact. Training consisted of a 1.5 hour classroom session covering the application domain (date definitions, codes, structures, organization, key actors, etc), two 1.5 hour classroom instruction sessions in the respective language followed by a paper and pencil test. Both treatment groups (i.e., SQL and USL)

did poorly in this first test. They were then given six 1.5 hour hands-on practice sessions with the system using requests modeled after actual user requests. An additional 1.5 hour classroom session was then given in each language followed by another six 1.5 hour practice sessions [5]. At this point subjects were given a second test to determine whether they possessed sufficient language skills to be able to answer user requests.

4.1 Method

The second test was constructed with questions that described problem situations in the application domain. Subjects were then asked to write one or more queries that would provide the information needed to answer the question. The questions were chosen to be similar to expected requests from real users. For example,

> Q4 - The Alumni Federation is preparing a letter to be sent only to alumni that have donated over $100,000 in 1981. Obtain the names, addresses, and the 1981 donations for those alumni.

The questions differed in their degree of difficulty and were placed in a constrained random order with an easy question first and last. The test contained thirteen questions and subjects were asked the time taken to answer each question as well as their perception of request clarity, complexity, and their certainty of a solution strategy. Care was taken not to bias the questions toward one or the other language. The test was administered in two mixed treatment groups. Written instructions were distributed with the test.

Test questions were scored on two difficulty scales, one a rank order and the other a four point scale ranging from simple to complex, by two graders familiar with the languages [6]. Question answers were scored, by two graders, on two scales, one an 11 point interval-level scale ranging from 0 to 10, the other an ordinal-level scale similar to the one used by Welty [Welty and Stemple 1981], which was based on a method used by Reisner (1977). The basis of scoring was the extent to which the answer would produce a correct result from the system. The Welty/Reisner method establishes categories of errors that are meaningful in the context of the experiment (such as those that are likely to be corrected by a good compiler) and makes a distinction between different levels of errors (minor and major). It has the advantage of more precisely linking query errors to scores than does an interval method. However, some of the categories do not apply to natural language (for example, operand errors) somewhat reducing the scheme's usefulness as a basis of comparison between languages.

Additionally, natural language answers were scored on their syntactical correctness, on their naturalness, and on whether there was a more direct (compact) way to express the query.

4.2 Results

The mean score on the test, using the interval scale, was 74.9 (n=8) with a standard deviation of 11.78 and a range of 62 to 89. There was no significant association between sex, years of programming experience, or number of computer languages known [7] and test score, although there was a positive association with age, GMAT score, years

of college, and years of work, which is to be expected. Since all subjects scored higher than the 50% level deemed acceptable to move on to the next stage of the field experiment, the training phase was considered successful.

The mean test score for SQL subjects was 75.19 (s.d.=7.75) while the mean score for USL subjects was 74.62 (s.d.=12.58), a difference that was not significant (T-Test t=.11, p=.470). However, the standard deviation for the USL subject scores was almost twice that of the SQL subjects suggesting more variation in USL subject performance. No difference between treatments was found in the distribution of average question scores (Figure 4).

One early hypothesis developed was that subjects using USL might perform better than those using SQL on easy and difficult questions, while SQL subjects might perform better on questions of medium difficulty. The rationale was that the formal structure of an artificial language might act as a barrier to expression when the problem was either very easy or very difficult. For example, with easy queries an artificial language may require key words at particular points and more tokens than natural language. Figure 5 is a plot of average question score vs. question complexity. Both languages exhibit a general downward slope of average question score as a function of question complexity, which would be expected, although this is only significant for USL. (correlations between average language question score and question complexity -- for USL: $r = -.791$, $p < .001$; for SQL: $r = -.457$. $p = .058$.)

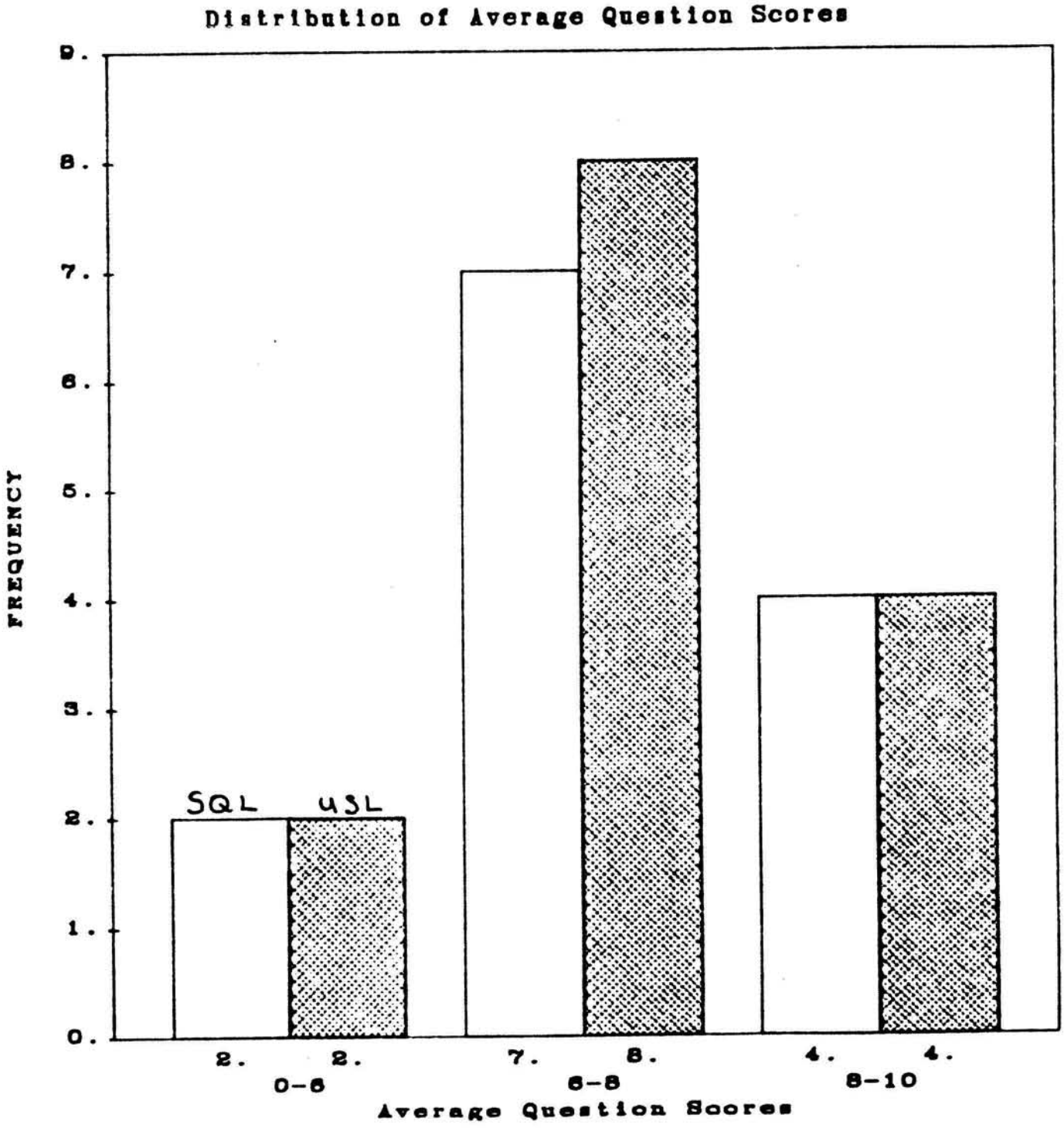

FIGURE 4

The questions were scored, for both languages, on a four point complexity scale to indicate how difficult the question was to answer in that language for the test application. Ranking the thirteen questions on the basis of this complexity score produced a correlation of rho = .62, p = .012 (spearman rho) between languages. This was expected because a particular question may be relatively easier to answer in one language than another which would alter the ranking. The two language complexity rankings were averaged to create a complexity index that would apply equally well to both languages and this was plotted against the difference in average language scores (Figure 5). Discounting the lowest average complexity question, the difference plot appears to have an inverted 'U' shape. However, only three of the differences in average language score are significant (Table 1), two favoring SQL (questions 1 and 4) and one favoring USL (question 3). There was no clear reason why these questions should be easier to answer in one language or the other. On the basis of these three questions there appears to be no simple relationship between average language scores and complexity.

No association was found between language type and demographic characteristics. The previously mentioned association between average score and years of college suggested that possibly the difference in average language score would be altered when controlled for years of college, but this was not the case.

In order to compare SQL performance with prior research, subjects were graded on a modified Welty category scale. As shown in Table 2, except for SQL subject 3, the paired interval and category scores are

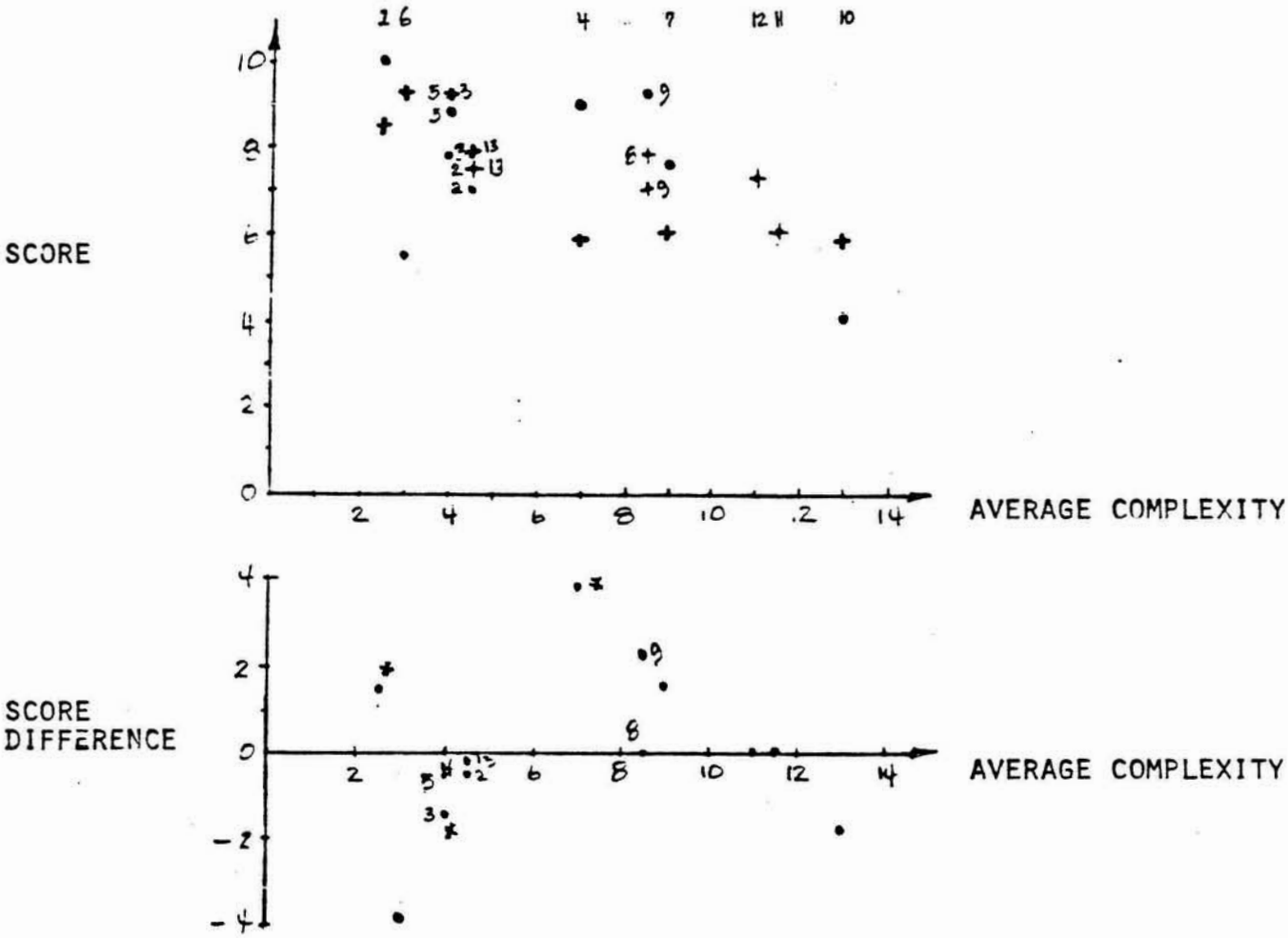

FIGURE 5

RELATIONSHIP BETWEEN AVERAGE TREATMENT SCORE AND AVERAGE QUESTION COMPLEXITY

• -SQL
+ -USL
1,2,etc.-Question Number
* -Difference Significant at the 0.05 Level or Better

TABLE 1

AVERAGE SQL AND USL ADVISOR GRADE BY QUESTION

QUESTION NUMBER	SQL $\bar{x}$	SQL s.d	USL $\bar{x}$	USL s.d	Δ SQL-USL	t	p	COMPLEXITY	COMMENTS
1	10.0	0.0	8.5	0.6	1.5	5.2	.007*	1	SQL +
2	7.0	2.5	7.5	2.4	-0.5	-0.2	.390	1	
3	7.8	1.0	9.2	1.0	-1.5	-2.2	.034*	2	USL +
4	9.5	0.6	5.8	2.6	3.5	2.8	.034*	2	SQL +
5	8.8	0.5	9.2	1.0	-0.5	0.9	.198	2	
6	5.5	4.1	9.2	1.0	-3.7	-1.8	.088	1	
7	7.4	2.4	6.0	2.9	1.5	0.8	.229	2	
8	7.8	1.0	7.8	2.5	000	000	1.000	3	
9	9.2	1.0	7.0	3.5	2.2	1.2	.150	1	
10	4.0	2.2	5.8	2.1	-1.8	-1.8	.143	3	
11	6.0	0.8	6.0	2.8	000	000	1.0	3	
12	7.2	2.5	7.2	3.1	000	000	1.0	4	
13	7.5	3.7	7.7	1.3	-0.1	-0.1	.453	2	
Grand Average	7.5	0.8	7.5	1.3	0.1	0.1	.470		
WAV	7.0	0.6	6.3	1.3	0.7	1.0	.194		

WAV -welty score average

SQL+-SQL significantly better than USL

USL+-USL significantly better than SQL

* -significant at the 0.05 level or better

TABLE 2

CORRELATION BETWEEN INTERVAL AND CATEGORY SCORES

SUBJECT	TREATMENT SQL	USL
1	.940(p=.001)	.768(p=.001)
2	.910(p=.001)	.912(p=.001)
3	.493(p=.004)	.957(p=.001)
4	.944(p=.001)	.910(p=.001)
AVERAGE	.965(p=.000)	

Interval score: ten point scale with 10 high

Category score: modified Welty scale

p is the Pearson product moment correlation coefficient

highly correlated. The correlation between average SQL interval score and average SQL category scores (modified Welty) is .965 ($p<.001$) suggesting little difference between the scoring methods. Using the category scores, treatment means are SQL, 69.81 (s.d.=17.72) and USL, 65.19 (s.d.=10.92); there still is still no significant difference between them (T-Test t=.86, p=.205).

Because USL has restrictions that make it appear different from correct English, we were interested in determining how natural USL queries were. Subject's USL queries were rated for naturalness and English syntactical correctness, on a three point scale, by a person unfamilar with USL syntax. The mean value of query naturalness was 1.9 (with 3 being natural) while the mean value of query English syntactical correctness was 2.3 (with 3 being correct). Correct USL queries were constructed and subjected to the same rating scheme. Naturalness and correct English syntax for correct USL queries were 2.2 and 2.8 respectively. These results suggest that even correct USL differs considerably from English, especially in naturalness.

4.3 Limitations

These early laboratory results are based on data from only eight subjects. Measures and techniques have not been finalized (for example, naturalness, correct English syntax, and complexity). We still are learning how to develop tests that are not biased toward one language and to standardize our training methods. The field study on which these results are based is currently underway; later experiments may use different techniques and we may modify the

conclusions of this early work. Finally, the prototype Natural Language system USL, upon which this research is based is continually evolving so that certain deficiencies become remedied as the project progresses.

4.4 Implications Of Findings

The finding that all subjects scored higher on the test than the cut off grade (50%) suggests that both languages can be learned with a combination of instruction and practice. However, although we use the same training approach used by Reisner (1977) and Welty and Stemple (1981), we were not as successful with our initial classroom work as they were. The overall result of the training process, though, does appear comparable. Using the same scoring method (mean percentage of essentially correct scores) as Welty, our SQL treatment subject test scores are similar to those found by Welty [Welty and Stemple 1981] and [Reisner 1977]. Welty's SQL subjects (two tests, n=35 and n=39) had an essentially correct answer percentage of 67.0 and 59.5 on twenty questions of varying degrees of difficulty. This compares with our average essentially correct SQL subject score of 76.9 on thirteen questions of varying difficulty. Welty also found that subjects scored higher on easier questions than they did on more difficult ones, a result we also observed. In an earlier study similar to Welty's, Reisner's SQL subjects had a percentage of essentially correct scores of 72 (n=64) using roughly the same scoring approach. It is reasonable to expect that our subjects would score higher than either of the two previous experiments since our procedure included

hands-on experience and involved more contact time. Considering differences in subjects, training methods, material and time, and test content, the results of these studies are quite consistent.

For our level of subject training there appears to be little difference in performance between SQL and USL as measured by test scores. Differences in performance found on certain questions suggest that one language may be better suited to certain functions than the other. Although it is too early in our research for us to come to any conclusion, we speculate that certain features of USL, such as counting, built in functions such as 'average', and the ability to easily create temporary variables may be easier to use than SQL for certain retrievals. Correspondingly, we feel that the difficulty in controlling output formats may give SQL an advantage in certain other situations.

There appears to be no simple relationship between differences in language performance and question complexity. We suspect that when the features of a languages maps well into the requirements of the question, performance improves (for example, when a qualification such as 'Italian' as in 'the Italian alumni' has been defined in Natural Language) but we have yet to identify the conditions under which this occurs. In order to more fully explore this issue we are in the process of defining four levels of complexity: surface, language, underlying database structure, and computational. It may be that by disaggregating complexity (particularly the second and third categories), a clearer relationship with performance will emerge.

Correct answers in USL appear to be different from English, partially explaining poor USL performance without training. We agree with Shneiderman's (1981) notion that restrictions in practical Natural Language Question-Answering systems would have to be learned, thus negating one of the major advantages of Natural Language. Whether subjects retain SQL better than USL, because of, possibly, less proactive interference, or a more formal structure, remains to be investigated.

We view the results of this laboratory study as a performance upper bound. That is, in real applications we would expect other factors, such as system loading, database size and complexity, operating system environment, the extent of networking, line condition, and terminal type to reduce performance below what we and other researchers have observed in laboratory experiments. On the other hand if a system provides constructive feedback to subjects, then learning may take place which could improve performance over that found in a laboratory setting. Our intuitive feeling at this point that the field results will be poorer than what we have found in the laboratory study. This position differs from Krause (1979) who suggests that field studies should perform better than laboratory studies. Subjecting this notion to an empirical test is one of the objectives of the field experiment.

5.0 STATUS OF THE FIELD EXPERIMENT

As of May 1982, the field experiment is in the second measurement period shown in Figure 3 (arrow over the second pair of x4s). The subjects have been crossed, trained, retested, and have begun to serve principals with their new language. Although we have not completed analysis of the first measurement period data we do have some preliminary general observations.

5.1 Training

We did have to train natural language subjects in a variety of material which was not anticipated. First, we had to describe the philosophy and coverage of the language. This was done by emphasizing restrictions. Then, we had to show subjects substitute procedures for getting around language 'gaps'. Third, we had to describe the meaning of messages and other feedback from the system. Much of this material is likely to be difficult to retain. There are no easy rules to guide users as there are with some artificial languages. It had not been obvious to us at the beginning of the project what topics would be important for natural language users to learn or how to go about teaching them.

Both treatment groups had to learn the mechanics of logging on to the system and the various categories of operating system messages and their meanings. Both groups had to be given training in the application system data meanings and codes. Some communications problems arose between advisors (subjects) and principals, because

principals had developed their own short hand form of reference, for instance, refering to a certain group of prospects as 'ones' (giving more than $500) because they have that code in reports. SQL subjects had to be trained in the data structure, SQL syntax, and functions.

5.2 Application Domain.

We expected that principals (as opposed to laboratory subjects) would be familiar with the details of their application domain down to data codings and structure. What we found out, instead, was that each principal developed their own terminology which others learned to interpret in communicating with them. These dialects were not directly related to the format or contents of the actual data. Nor did principals have the time or desire to get into the details of the data. A 'gate keeper' maintained the data, resolving inconsistencies, interpreting information requests, and controlling access. Most of this was transparent to principals. Therefore moving principals closer to their data (by giving them languages for direct database access) presents a new set of problems for them.

We also observed that principals tended to use a word in different ways. For example, both

Donors give contributions.

and

Schools give degrees.

Because of the limitation in USL that a word (in a particular grammatical role) can only have one meaning [8], the application designer is placed in a difficult position of having to chose one or another definitions of 'give'. Users may consider the requirement that words be used consistently, as a major restriction. While one solution may be to give each user his own application lexicon, we wonder who will configure these lexicons and who will do the analysis on which the configuration is based? Inexperienced users may not be sufficiently skilled in the system to perform these activities nor may they be willing to take time away from their normal activities. Eventually it may be possible to automate the development of user lexicons so that principals can easily tailor their own application systems.

5.3 Application Development

Designing (configuring) a natural language application differs from current database design practice in a number of ways. It is necessary to understand just how a principal uses language - what are the concepts and how are they related to actual data. This is another level of analysis required in design. In addition to the normal database design activities of data definition and structuring, one must perform an analysis of queries and define the linguistic elements to be represented in the system. In normal database design, one usually seeks a balance between efficiency of retrieval and update. In USL, query and terminology analysis is a starting point for design; those queries that cannot be handled by the system require additional

concepts to be defined. Consequently, the design process becomes more iterative, takes place over a long time interval, and requires actual used participation, which they may be difficult to obtain since users may not be receiving useful output during this period. It is not clear whether design closure will ever be reached. Also, the resulting database structure may be different than that prescribed by database design requirements alone.

The following example is illustrative of the difficulties we encountered during design. One query:

List the Puerto Rican Alumni

could not be interpreted because the adjective 'Puerto Rican' had not been defined as a view in the application lexicon, despite the conviction of the application designer that there was a view for each country in the database. Further investigation revealed that - in the application lexicon - the view Puerto Rican had been defined as a 'country' with a code value of 'PR' while in the database Puerto Rico was represented with a code of 'PR' in the 'state' field. This illustrates the type of misunderstandings which are only found after extensive system use.

It is sufficient to say that the design process is different than that currently followed for database application development and that a new set of design tools will be needed. Our findings with respect to application development will be discussed in a forthcoming paper.

5.4 Importance Of The Interface

We have become acutely aware of the importance of the total interface to the user. The weakest link of the interface appears to mask the other parts of the system, because the user concentrates on resolving the problem that is blocking him from accomplishing work. Frequently such things as terminal types, telephone lines, and operating system elements are considered beyond the domain of the application system designer. This is particularly true of large multi-user systems with complex operating systems where users are often left to fend for themselves outside of the target system. We believe it is important for designers and evaluators of application systems to take a broader view that includes the total environment. It is difficult to separate problems inherent in the basic concept of a system from those that are attributable to a poor user interface.

5.5 Importance Of Feedback

Feedback from the system, usually in the form of error messages, is critical for a user to achieve any reasonable level of performance and for learning. Feedback is needed to understand the consequences of actions. Without feedback, a user must possess much more knowledge of system operation in order to predict future system states which defeats the purpose of interface languages for novice users. Without feedback it is not possible to know the current system state. Confusing or non-existing feedback misleads a user and produces different problem solving behavior than a system with useful messages. USL currently has poor error messages and little user feedback. This,

more than any other factor, compromises our ability to evaluate the basic system concept (i.e Natural Language).

6.0 CONCLUSION

Although we have yet to complete the field evaluation much has been learned. An approach to application evaluation that can be used in many settings has been developed. The power of coordinated laboratory studies and field evaluations to complement and reinforce each other have been demonstrated.

The present plan is to remove as many of the sources of error as possible so that subjects expend more of their energy in the target languages and less on the surrounding environment. The most important question is whether a language, with certain features and structure, influences subjects' problem solving behavior and performance.

Our observations of the field experiment, although preliminary and incomplete lead us to question the notion that 'end users' will make effective use of computer systems if only it were easier to learn an interface language. While natural language systems, as a class of interface languages, appear to have a future, the principals in this application had little patience for the quantity and diversity of problems encountered by the subjects. Although there were difficulties in the field setting, the performance of USL in the laboratory experiment is encouraging. We believe that further experimentation is warranted.

Acknowledgements

This work was carried out as part of a joint study with the IBM Information Systems Group in White Plains, N. Y. The assistance of John Kesick and Ken Michielsen is appreciated. We are indebted to our principal users, Ann Dauberman, Associate Director of External Affairs, and Jay Avelino, Director of the Graduate School of Business Administration Alumni Association for their support and time, and to Joe Ciciollo of the Administrative Data Processing Center for his assistance.

The comments of Dr. Judith Reitman on a previous version of this paper are greatly appreciated.

Footnotes

[1] - Most of these methodologies aim at reducing maintenance costs by producing systems with fewer problems and with more standardized code than traditional, 'ad hoc' methods. What ever productivity gain may have resulted has been more than offset by increases in labor cost.

[2] - Exceptions are LSNLIS system (Woods et al., 1972) and REL (Thompson and Thompson, 1975) as well as the previously mentioned field evaluations of USL (Krause, 1979) and TQA (Damerau, 1979).

[3] - USL was originally developed with a German grammar that has since been extended to English, Dutch, and Spanish.

[4] - See "Diagram: A Grammar for Dialogues" (Robinson, 1982) for an interesting description of a phrase-structured natural language system with intersentential reference.

[5] - The first practice sequence was interrupted by Christmas vacation, final exams, and Semester Break covering a period of about six weeks. Therefore, the amount of contact to acquire skill is misleading since it includes a re-learning component.

[6] - One of the graders was the person who made up the questions.

[7] - An attempt was made to control for demographic variables by selecting subjects with the same background. Thus, many of the demographic variables show little variation.

[8] - To be precise, more than one meanings may be given to a word in USL, but there is no mechanism of disambiguation according to context. Thus, in our case, each use of the verb 'to give' will result in two interpretations. one of which will be wrong and possibly confusing. In addition, this presents an obvious system inefficiency.

References

1. Alloway, R.M. and Quillard, J.A., "User-Managers Systems Needs," CISR WP 86, Sloan School of Management, MIT, Cambridge, Mass., (1982).

2. Artificial Intelligence Corp., "Intellect Query System", Reference Manual, (1982).

3. Astrahan, M.M., and Chamberlin, D.D., 'Implementation of a Structured English Query Language',Communications, ACM, Vol 18, No 10, October (1975), 580-588.

4. Bertrand, O., J. J. Daudennarde, D. Starynkevitch, and A. Stembock-Sermor, "User Applications Generator," Proceedings of the IBM International Technical Conference on Relational Data Base Systems, Bari, Italy, p. 83., (1976).

5. Computerworld, Nov. 30, p.29., (1981).

6. Damerau, F. J., "The Transformational Question Answering (TQA) System: Description, Operating Experience, and Implications," Proc. segunda conferencia internacional sobre bases de datos en humanidades y ciencias sociales, Madrid, facultad de informatica, (1980).

7. Damerau, F.J., "The Transformational Question Answering (TQA) System Operating Statistics", IBM Research Report, RC 7739, (1979).

8. Halpern, Mark, "Foundations of the Case for Natural Language Programming," IEEE Spectrum, March, p. 140-9., (1966).

9. Kay, M., "Experiments with a Powerful Parser," Proceedings of the Second International Conference on Computational Linguistics, Grenoble., (1967).

10. Krause, J., "Preliminary Results of a User Study with the 'User Specialty Languages' System, and Consequences for the Architecture of Natural Language Interfaces", IBM Heidelberg Scientific Center TR 79.04.003, (1979).

11. Lehmann, H. 'Interpretation of Natural Language in an Information System', IBM Journal of Research and Development, vol.22, no.5, September, (1978).

12. Lehmann, H., Ott, N., and M. Zoeppritz, "User Experiments with Natural Language for Data Base Access", Proceedings of 7th International Conference on Computational Linguistics, Bergen, 1978.

13. Malhotra, A, J. C. Thomas, J. M. Carroll, and L. A. Miller, "Cognitive Processes in Design," Int. J. Man-Machine Studies, 12, p. 119-40., (1980).

14. Malhotra, A. and I. Wladawsky, "The Utility of Natural Language Systems," Research Report #RE5739, IBM T. J. Watson Research Center, Yorktown Heights, NY, (1975).

15. Miller, L.A., 'Natural Language Programming: Styles, Strategies and Contrasts', IBM Systems Journal, vol.20, no.2, (1981).

16. Ott, N. and M. Zoeppritz, "USL - An Experimental Information System Based on Natural Language," in L. Bolc ed. Natural Language Based Computer Systems, Macmillan, London., (1979).

17. Petrick, S.R., "On Natural Language Based Computer System", IBM Journal of Research and Development, 20, 4, July, 1976.

18. Reisner, P., "Use of Psychological Experimentation as an Aid to Development of a Query Language", IEEE Transactions of Software Engineering, SE-3, 3, (1977), pp.218-229.

19. Robinson, J. J., "DIAGRAM: A Grammar for Dialogues," Comm. ACM, 25:1 (January), p. 27-47., 1982.

20. Shneiderman, B., Software Psychology, Withrop, Cambridge/Mass., (1980).

21. Simmons, R. F., "Natural Language Question-Answering Systems: 1969," Comm. ACM, 13:1 (January), p. 13-30., 1970.

22. Stohr, E.A, J.A.Turner, Y.Vassiliou, N.H.White, "Research in Natural Language Retrieval Systems", 15th Ann.Hawaii Int.Conf. on System Sciences, Hawaii, (1982).

23. Tennant, H., "Experience with the Evaluation of Natural Language Question Answerers," Working Paper #18, Advanced Automation Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Il., (1979).

24. Thomas, J.C. and Gould, J.D., "A Psychological Study of Query by Example", Proceedings of NCC, Vol. 44, (1975).

25. Thompson, F. B., and B. H. Thompson, "Practical Natural Language Processing: The REL System as a Prototype," in Advances in Computers 13, M. Rubinoff and M. C. Yovitz, ed. Academic Press, New York., 1975.

26. Vassiliou, Y., and Jarke, M., "Query Languages - A Taxonomy", in Human Factors and Interactive Computer Systems, Y. Vassiliou (ed), ABLEX, Norwood, NJ, 1983.

27. Welty, C., Stemple, D.W., "Human Factors Comparison of a Procedural and a Non-Procedural Query Language", ACM Transactions on Database Systems, (1981).

28. Woods, W.A., Kaplan, R.M., and Nash-Webber, B., "The Lunar Sciences Natural Language Information System", Bolt Beranek and Newman, Cambridge, Mass, June, (1972).