A FIELD EVALUATION OF NATURAL LANGUAGE FOR DATA RETRIEVAL

Matthias Jarke
Jon A. Turner
Edward A. Stohr
Yannis Vassiliou
Norman H. White
Ken Michielsen

November 1983

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

Working Paper Series

CRIS #62

GBA #84-8(CR)

Published in IEEE Transactions on Software Engineering, Vol.SE-11, January 1985.

A FIELD EVALUATION OF NATURAL LANGUAGE FOR DATA RETRIEVAL

Abstract

Although a large number of natural language database interfaces have been developed, there have been few empirical studies of their practical usefulness. This paper presents the design and results of a field evaluation of a natural language system - NLS - used for data retrieval.

A balanced, multifactorial design comparing NLS with a reference retrieval language, SQL, is described. The data are analyzed on two levels: work task (n=87) and query (n=1081). SQL performed better than NLS on a variety of measures, but NLS required less effort to use. Subjects performed much poorer than expected based on the results of laboratory studies. This finding is attributed to the complexity of the field setting and to optimism in grading laboratory experiments.

The methodology developed for studying computer languages in real work settings was successful in consistently measuring differences in treatments over a variety of conditions.

Acknowledgments

This work was carried out as part of a joint study with the IBM Information Systems Group in White Plains, NY. The assistance of our principals, Ann Dauberman and Jay Avelino of NYU, is greatly appreciated.

1.0 INTRODUCTION

Although a large number of natural language understanding systems have been developed, their practical feasibility and desirability is still unproven. Unfortunately, few systems have been subjected to rigorous empirical studies. Many claims of the various approaches must thus remain unresolved. The systems that have reached the highest degree of maturity are based on linguistic concepts without much recourse to knowledge-based techniques. The Advanced Language Project at NYU attempted a comprehensive laboratory and field evaluation of such a restricted natural language front end, called NLS, to a relational data base system. NLS is a general purpose data base query language that uses a bottom-up parser, an English grammar consisting of some 800 BNF rules, an application specific lexicon, a set of interpretation routines for semantic analysis, and a relational data base management system for data retrieval [9].

Together with two laboratory experiments [20], [22], a field study spanning approximately half a year constituted the primary strategy for evaluating NLS. The objective of the field study was to investigate the problem-solving performance of NLS in a real-world, yet partially controlled setting.

Can subjects who have real work to do make use of a natural language application? Assuming that a satisfactory natural language application can be designed, under what circumstances will it be superior to a structured query language? Finally, what is the interplay between subjects' problem solving behavior and the features of the application languages?

It is generally presumed that the need to learn the syntax and semantics of an artificial computer interface language acts as a barrier for the novice or infrequent user of an application system [10]. One strategy for dealing with this problem is to provide these users with a natural language interface. However, due to limitations in building these interfaces, such systems all have restrictions of one form or another (e.g., limitations in inter-sentential reference, pronoun references, ellipsis, or coordination, etc.). Thus, what is really being investigated is the extent to which restrictions (characteristics) of a particular system influence how subjects use that system rather than how they use 'pure' natural language (that is, communication with a human in native tongue). Consequently, the degree to which any evaluation study of a particular system can be generalized is open to question. In spite of this limitation it is believed that a great deal can be learned from evaluations of specific languages.

While there have been a number of laboratory studies of implemented artificial or natural language interfaces (e.g., [4], [13], [14], [23]) there have been relatively few field studies of these systems (for exceptions see Krause's field study of NLS [7], [8], and [2], [3], [5]). Theoretical and empirical research in the evaluation of natural language systems is reviewed in [8], [18], [20]. As Tennant [19] observes, the lack of evaluation studies in real field settings has left several critical questions about these systems unanswered. As a result, little methodology has been developed for performing field evaluations. Consequently, the rationale for the experimental design of a field study to evaluate a natural language interface becomes of particular interest in and of itself. It is also evident that the results of a field study will be influenced greatly by the specifics of the interface languages selected for evaluation, the application area in which the evaluation takes place, and the subjects themselves.

This paper first presents highlights of the experimental design and summarizes the multi-level evaluation scheme used to capture information from subjects using the two alternative languages. The results of an analysis of the experimental data are then presented, followed by a discussion of the implications of these results for the design of natural language systems and for evaluation research.

2.0 EXPERIMENTAL DESIGN

In this section, the main experimental design decisions made for the field study are reviewed. As Petrick [12] and Simmons [16] observe, question-answering systems (where natural language questions are transformed into formal language queries by syntactic and semantic analysis) are a likely use of a natural language interface largely because the target data base tends to limit and clarify the domain of discourse. Consequently, a data base query system is a reasonable application of natural language that has broad utility. The following conditions were established for the field study:

1. Subjects had to be performing real work. This required the design of a non-trivial application system for a work setting.

2. Subjects should approximate the characteristics of young professionals, a group with sufficient application domain knowledge and analytic skills to be likely to use a natural language data base interface.

3. A frame of reference should be established in which results could be interpreted. This implies developing a formal evaluation scheme.

5. As many controls as possible should be established. The major difficulty with field studies is attributing differences in outcomes to differences in treatments. Thus controlling for unexpected factors is one of the most important experimental design issues.

These objectives were met in the following way.

2.1 Application In Alumni Administration

The application selected, a question-answering system about alumni of the Graduate School of Business Administration (GBA) at New York University, maintains demographic and gift history data of school alumni, foundations, other organizations, and individuals. The school has over 20,000 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 donations are submitted to the Associate Director for External Affairs from faculty, the Deans, student groups and other parties concerned with fund raising or alumni relations at the school. Either the Associate Director has the information or she calls the school representative at the Alumni Federation. Periodically, the representative produces reports from a large batch transaction processing system that serves all of the Schools of the University and returns them to the Associate Director.

Data for the natural language application was extracted from the University's record keeping system and used to load the NLS data base. Since the NLS was used strictly for querying, the data base was refreshed with extractions from the University's system. The application as implemented for the field experiment contains four relations:

Prospect Master	- name, id, demographic data	- 25,000 tuples
Gift Summary	- id, gift 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 2.1 shows the database structure as an entity-relationship diagram. The domain of discourse includes alumni and non-alumni who have given to the school, their gift histories, their education, their demographic data, and their roles as solicitors and for matching gifts.

Place Figure 2.1 about here

Figure 2.2 presents a simple example of a task and its solution in NLS and SQL. In this case, only one query is required, but more complex tasks can require up to about 20 queries for their solution.

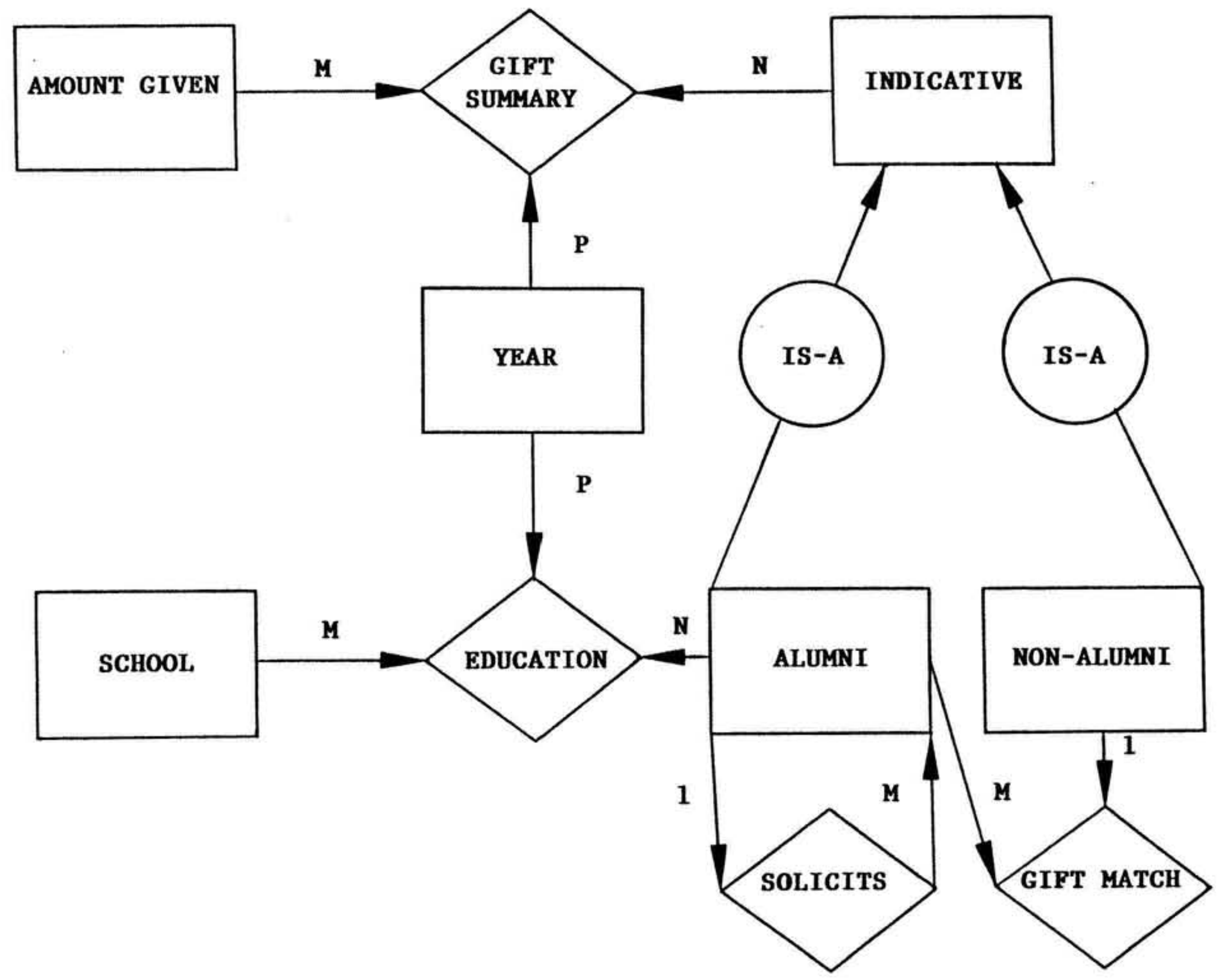

Figure 2.1: Entity-Relationship Diagram of the Alumni Application

A TASK DESCRIPTION

A list of alumni in the state of California has been requested. The request applies to those alumni whose last name starts with an "S". Obtain such a list containing last names and first names.

NLS SOLUTION

What are the last names and first names of all California Alumni whose last name is like S% ?

SQL SOLUTION

```
Select lastname, firstname
From   donors
Where  srccode = 'al' and state = 'ca'
                      and lastname like 's%';
```

Figure 2.2: Example of a Simple Task and its Solution in Both Treatments

2.2 Paid Intermediaries

Initially it was thought that Deans and Development Officers would directly use the system. However, it quickly became apparent that these principals did not have the time or the patience to participate in a research project. Also, the system would have only two regular users, too few for statiscally valid results.

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 (termed "advisors") on behalf of principals. Subjects would meet with principals and obtain a verbal information request: their task. They would then interact with the system to obtain an answer, by typing in one or more queries in the retrieval language they were using. The answer to the task (some combination of the answers to queries) would be returned by the intermediary 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 NLS and a research project.

2.3 Comparative Study

In field studies, the challenging issue is to control for factors not directly measured. It is difficult, especially in exploratory studies, to anticipate what factors will influence outcome variables. Rather than attempting 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. In this way the differential in a parameter, rather than its absolute value, becomes an important factor.

In order to allow for a fair language comparison and to reduce the influence of factors outside the languages to be compared, such a reference language should: (1) be directed towards the same type of users as the natural language interface, e.g.,novice users; (2) work in a similar system environment; (3) have been subject to previous studies so it may be used as a point of reference to interpret the study results.

SQL was selected as the reference language. SQL had been extensively studied [4], [13], [14], [17], [23], both query systems used the same underlying data base management system with the result that one application data base could support both applications, and NLS mapped queries into SQL promoting comparative analysis (e.g., complexity analysis, see section 3).

2.4 Counter-Balanced And Paired Treatment Design

Because other researchers had found performance among individuals to be highly variable, a counter-balanced design was selected that would enable between-group contrasts to be verified by within-group contrasts. Figure 2-3 shows the research design. Subjects were divided into two treatment groups, Group 1 and Group 2, both of which were trained in the application domain, and then in either NLS or SQL. They were tested, and following that, they interacted with principals (phase 1). At the end of an approximately six week period, treatments were crossed: group 1 was given the second language (SQL) and group 2 was given the first language (NLS). They were trained in the new language, tested and then interacted with principals (phase 2). At the end of another measurement period subjects were given a practice session and then used whichever treatment they preferred to accomplish selective tasks (phase 3). The research design was intended to reflect the regular, but infrequent, use of an application system that might be typical of novice or specialist professional users [6], [21].

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

X1	- Application Training
X2	- NLS 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 2.3: Multi-Factoral, Repeated Measure, Balanced Design For The Comparative Evaluation of Natural Language Question-Answering Systems

----

Place Figure 2.3 about here

2.5 Selection And Comparability Of Subjects

Advertisements were posted at the Graduate School of Business Administration and at the College of Business and Public Administration (undergraduate) 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 were 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 familiar with computing, but not enough to be an expert. The most experienced subject had written 15 BASIC programs and had minor familiarity (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. It is believed that 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.

2.6 Training Of Subjects

Subjects were trained using a combination of classroom and hands on practice sessions. Classroom sessions for SQL were modeled after those used by Reisner [13] and Welty and Stemple [23]. NLS training concentrated on the underlying philosophy of NLS (i.e., no domain knowledge, non-AI based) by identifying the restrictions of the language, strategies for circumventing restrictions, language features that were not operational, and practice problems. In addition, both groups received training in the application area and were provided with a data dictionary. Prior to the beginning of the field experiment, both groups were given a paper and pencil test to insure that each subject had obtained an acceptable level of proficiency [20].

2.7 Hypotheses

Based on the results of the first laboratory experiment [20] and expectations from prior research, a set of hypotheses was formulated for the field study as follows.

> H1: There will be no difference in performance between subjects using the restricted natural language interface (NLS) and those using the more structured interface (SQL).

While an argument could be made that it is harder to learn the syntax of a formal language than it is to learn the restrictions of a natural language, an equally good argument could be made for the reverse, that it is harder to learn the many restrictions of a 'semi-natural' language due to pro-active interference [15], than it is to learn the syntax of a formal language. There appeared to be no compelling reason to favor one or the other of these positions. Furthermore, the results of the first laboratory experiment (a paper and pencil test) indicated no significant difference in performance between treatment groups.

H2: Subjects using NLS will be more efficient than subjects using SQL.

Efficiency is defined as the amount of effort required to use a language interface to accomplish a task. Artificial languages permit efficient expression because they can omit redundent information necessay in natural language. Yet, the rigid syntactic structure of SQL compared with the relatively compact expression possible in restricted natural language suggested that NLS subjects would expend less effort in doing their work. This is consistent with the results of the first laboratory experiment.

H3: The performance of subjects will be negatively related to the difficulty of the task they are attempting to accomplish.

Difficult tasks will require longer time for thought and subjects will be more likely to make errors requiring additional work.

H4: The performance of subjects will be negatively related to their perceptions of task difficulty and positively related to their understanding of a solution strategy.

Previous research [11] has shown a positive relationship between the perceptions a subject has about a task and their performance. It is reasonable to expect that subjects who perceive a task to be less difficult than other subjects will perform better on that task.

3.0 MEASUREMENT AND CODING

In this section, the evaluation criteria for the field experiment are described. The source documents for most of the coding were either forms filled out by subjects at the time they were working with the system, or hard copy computer session logs. Coding was performed by one of the investigators together with assistants and was verified in a number of ways including computer-assisted consistency checking.

3.1 Evaluation Objectives

In the field experiment, subjects were given tasks, by principals, to be accomplished by issuing a number of queries to a database. To represent this process, a hierarchical model of task accomplishment was developed. An evaluation scheme based solely on the correctness of individual queries, as often used in laboratory experiments, could be misleading in interpreting performance at the task (request) level. That is, a large proportion of correct queries does not necessarily mean that the task was successfully accomplished. Thus, a hierarchy of related coding schemes was needed - one at the task level and one at the query level.

The objectives of both coding schemes can be summarized as follows:

1. Measure the success of subjects in performing their task or sub-task. Success is based on both the syntactical correctness of queries submitted as well as the contribution of the answer towards accomplishing the overall task.

2. Measure the effort involved in accomplishing the task or sub-task.

3. Measure the factors that are likely to influence success and/or effort.

4. Capture subjects' perceptions about a treatment.

3.2 Definitions And Criterion Hierarchy

In addition to the task and the query levels of coding results, two other levels, the request and the session level complete the measurement approach (see figure 3.1).

Place Figure 3.1 about here

A request is a task description given by a principal to a subject. The answer to a request is a collection of database output which a principal can use to derive support for his or her decisions. On the request level, language-independent descriptive measures can be applied which are based on a conceptual model of the application. For example, a measure of request complexity was developed based on the number of entity types referenced in a request (see Figure 2.1).

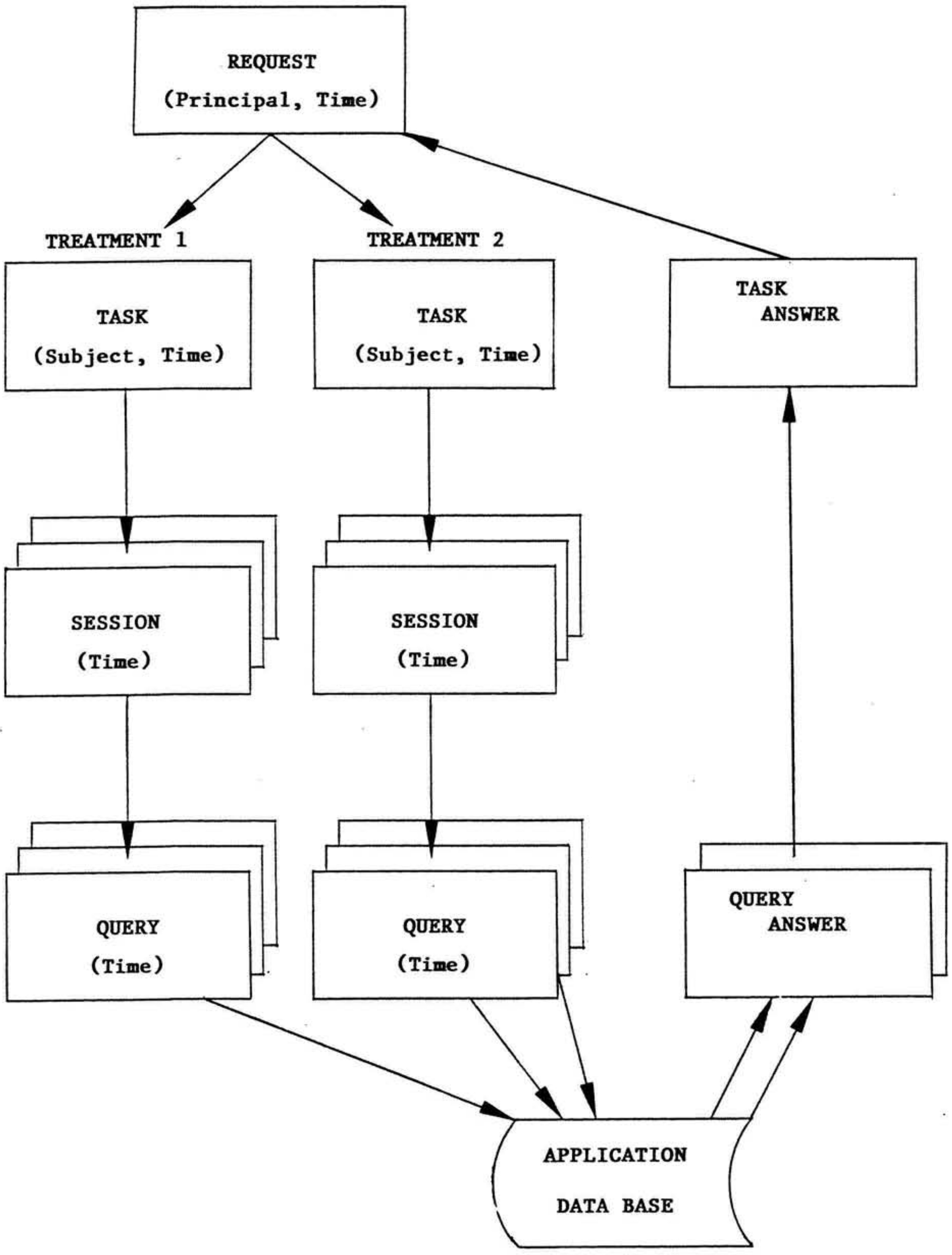

Figure 3.1: Evaluation Hierarchy for the Field Study

Each request was given to one or more subjects as <u>tasks</u> to be solved using a specified treatment (language). Most requests were given to at least one subject from each of the two treatment groups. Such tasks are considered <u>paired</u>. Measures at the task level are language independent or language-dependent descriptions of the overall performance of the query language as a data base accessing tool. Subjects recorded task content and perceptions about tasks on forms completed just after the task was assigned.

A subject could work on a task during one or more continuous periods of interaction with the system, called <u>sessions</u>. Measures at this level consist of subjective perceptions as well as the actual status of the system (e.g., system load, communications problems, duration of interaction). The contribution of a session to the overall success of a task was also captured.

During a session, subjects submitted one or more <u>query</u> (attempts) to the system. The query level permits a detailed analysis of the problem solving strategies of subjects and an evaluation of the adequacy of particular language features.

In addition to measuring inputs to the system (the task - query chain) the output of the system (response) must also be captured. Working from more detailed to higher levels, response measures can be developed at the query, session, and task level and they may consist of coding of the results of outcomes, likely problem sources, and subjective perceptions.

3.3 Measurement Strategy

Table 3.1 provides an overview of the measurement strategy used in the field experiment. Appropriate identifiers were used at all levels of measurement. At the task level, effort was captured by measuring the <u>number</u> of sessions, the length of <u>time</u> taken, and the <u>number</u> of queries used. The <u>complexity</u> and <u>uniqueness</u> of the task were used as control parameters. Success was measured by assessing task objectives and comparing them to actual results (outcome <u>correctness</u>). Likely <u>reasons</u> blocking task accomplishment were identified.

	CODE LEVEL			
CODE TYPE	request	task	session	query
identification	REQNO	ADVIS RLANG	SESNO	QNO

factors/
complexity/
perceptions
	RENETT
	RRELA
	RATTR
		RQUER(i)
		RPURP(i)
		RCLAR(i)
		RSTRA(i)
		RCOMP(i)
		RSTAN(i)
			SRQUER
			SRPURP(u)
			SRCLAR(u)
			SRSTRA(u)
			SRCOMP(u)
			SRSTAN(u)
				QRPR(i)
				QCHNG(i)
				QOBJ(i)
				QVARS
				QRESTR
				QJOINS
				QTATTR

effort			STIME SRUSLS(u)	QLGT

success		RPATH(i)	SRPATH(i)	QPATH(i)
				QOQUAL(i)
				QGRADE(i)

problem		RPSRC(i)	SRPSRC(i)
			SRUSPR(u)
				QINTBY(i)
				QERR(i)
				QPSOURCE(i)
				LPSRC(i)

(u) - code value determined by the user
(i) - code value determined by the investigators
Unmarked codes are identification codes or objective measures.
Definitions of all code names are given in Appendices A and B.

Table 3.1

Overview of the coding scheme

Task difficulty was considered a multidimensional concept and hence measured by several different factors. One factor involves identifying the number of <u>concepts</u> referred to in a request, that is, entities and relationships (refer to Figure 2.1). This is a 'conceptual' level representation of task complexity involving the number of 'objects' that a person has to deal with in accomplishing the task. Another factor involves identifying the number of and difficulty of the <u>operations</u> that are implied in accomplishing a task.

This is an 'operational' level representation of complexity and is language (e.g., treatment) dependent. It represents the transformations that a person invokes in order to accomplish a task. The third component of task difficulty is the 'surface structure' or 'formulation complexity' of the task representing the actual 'work' performed on the data before output.

The details of the task level measurement scheme are presented in Appendix A.

At the query level, effort was measured by the length of queries and the number of queries used in task solution. The complexity of a query and the solution strategy used by the subject were captured as control parameters. Success was measured by whether the likely objective of the query was accomplished (outcome correctness). Likely reasons blocking query accomplishment were identified. Perceptions of subjects' clarity of their task and how certain they were of their solution strategy as well as the adequacy and their preference for a particular treatment were also captured.

The primary detailed unit of analysis is the individual query. The general notion is that subjects analyze tasks, breaking them into small pieces of work, or subtasks, which they then attempt to accomplish. Each query can be thought of as a representation of a subject's approach to performing a subtask. The response from the system, then, determines whether the subtask was properly accomplished.

If the subtask is successfully completed, then, the next subtask is attempted, and so on, until the whole task is completed. However, if the subtask is not successful, then the subject is faced with two courses of action: either to attempt to diagnose the problem which prevented subtask accomplishment and re-perform the subtask, or change the subtask sequence, substituting a new subtask for the one that was unsuccessful. Figure 3.2 shows the possible paths that a query may take.

Place Figure 3.2 about here

Again, as with the task level analysis, both the input to the system and the system's response have to be captured. While descriptors of the input are based on various measures of complexity, similar to those used at the task level, the conceptualization of subtask accomplishment suggests the need to have a rich and varied coding scheme for representing subtask outcomes (that is, the response from the system). The basic coding model captures two aspects of outcomes: the extent of <u>success</u>, and the reasons for and attribution of <u>failure</u>. Complicating matters, many outcomes are not independent of inputs.

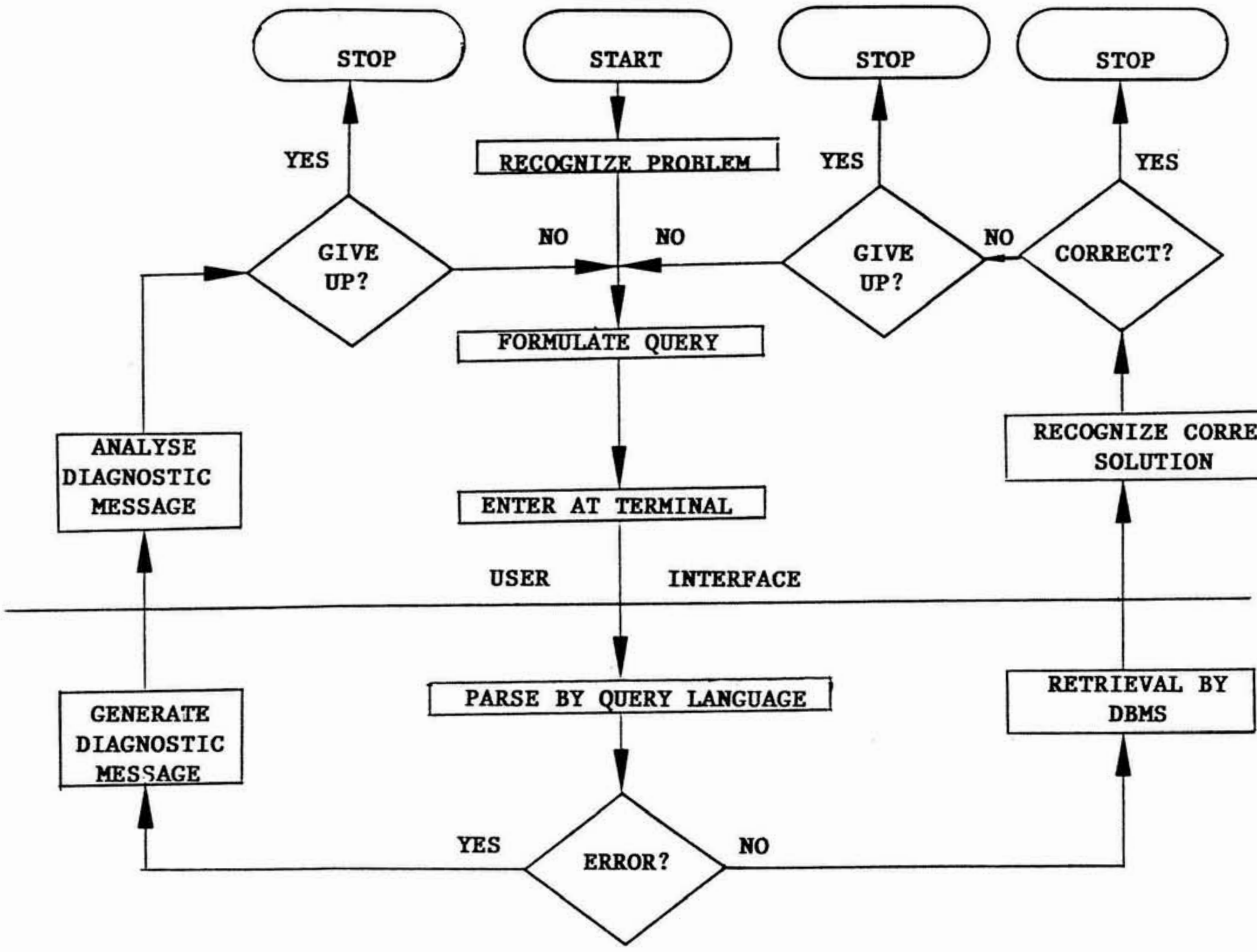

Figure 3.2: Simplified View of Query Evaluation and Error Sources

The complexity of developing a coding scheme for outcomes can be illustrated by the following categories of situations that must be differentiated.

1. A query may be syntactically correct, or it may have errors. Errors may be due to incorrect syntax or to a typing, spelling, or a communications interface error.

2. A syntactically correct query may still produce no or unusable output because of an incorrect reference or qualification (e.g., file name, index).

3. A syntactically correct query may still produce no or unusable output because of a semantic problem - it is the wrong question to ask.

4. A query may be both syntactically and semantically correct, but still return no (that is, null) output.

5. A query may be both syntactically and semantically correct, produce output, but the output may not substantially contribute to task accomplishment. This is particularly true when a subject attempts to test a language feature (possibly to gain confidence that the system still works), but that output has little to do with task accomplishment.

6. A query may be both syntactically and semantically correct, but does not produce output because of a system bug or a feature that does not work (for example, an inability to sort output).

7. A query may be both syntactically and semantically correct, but it may be canceled by a subject before it has completed execution (possibly, because it has taken too long, or it is estimated that it will take too long).

8. A query may be both syntactically and semantically correct, but produce only partial output. For example, the number of fields requested may exceed the space available on a page of output.

The details of the query level measurement scheme are presented in Appendix B at the end of this paper. A more detailed description of the complete coding scheme is given in ALP Technical Report No. 4, "Coding Schemes for the Field Experiments," available from the authors upon request.

4.0 EXPERIMENTAL RESULTS

4.1 Descriptive Statistics

Subjects were given 39 different requests to work on by principals during both phases of the experiment. This resulted in 87 tasks (request-subject pairs) being worked on by subjects. During 138 sessions, a total of 1081 queries were submitted to the system. Table 4.1 presents these global statistics by treatment - that is, by NLS or SQL - and by phase of the experiment. Although perfect pairing of requests was made impossible by scheduling problems, overall an almost equal distribution of work between treatments and phases was achieved.

	Phase 1	Phase 2	Total
Requests			
- total attempted	19	20	39
- paired (attempted in both languages)	12	16	28
Tasks			
- NLS	21	21	42
- SQL	25	20	45
total	46	41	87
Sessions			
- NLS	34	31	65
- SQL	42	31	73
total	76	62	138
Queries			
- NLS	343	313	656
- SQL	291	134	425
total	634	447	1081

Table 4.1

Descriptive Statistics for the Field Experiments

4.2 Task Level Analysis

On the task level, the analysis had to:

1. Control for differences in the difficulty of tasks (recognizing that perfect pairing of tasks would not be possible in all cases).

2. Establish the performance of subjects in terms of effort spent and final success in solving a task.

3. Identify the problems that prevented the maximum possible success from being achieved.

In the analyses that follow, distribution free (non-parametric) statistical tests are used for the most part, because the number of cases is often quite small and there is a possibility that a distribution may be skewed. The results of the distribution free statistics did not differ much from the corresponding parametric tests.

4.2.1 Difficulty Of Tasks -

As expected, most of the complexity measures were highly correlated. As shown in Table 4.2, a significant negative correlation was found between several of the complexity measures and request number suggesting that the requests apparently became easier during the latter portion of the field experiment. The number of entities (RENETT) and the complexity of required output operations (RATTR) were significantly less during the second second phase of the study (Table 4.2). However, the number of relationships (RRELA) and the number of necessary queries (RQUER) were not significantly different.

	Correlations REQNO - Request Number tau	number	p
RENETT - entities	-.260	87	.001
RRELA - relationships	-.175	87	.022
RATTR - rqd. output ops.	-.171	87	.023
RQUER - queries	-.095	87	.115

	Phase 1	Phase 2	F	P (t test)
RENETT - entities	4.1 (1.5)	3.2 (1.4)	8.42	.005
RRELA - relationships	2.2 (0.8)	1.9 (1.0)	2.58	.112
RATTR - rqd. output ops.	1.9 (0.9)	1.5 (0.7)	6.17	.015
RQUER - queries	4.1 (4.5)	3.4 (3.3)	0.69	.406

Table 4.2

Comparison of Complexity Measures Over Time

In terms of difficulty, as shown in table 4.3, SQL tasks were somewhat more complex than NLS tasks on one of the language independent measures (RENETT - number of entities) and on one of the perceived measures (RCLAR - task clairity), but these differences are not significant.

evaluation criterion	code	NLS	SQL	n	p (t test)
TASK COMPLEXITY MEASURES					
- no. of entities	RENETT	3.4 (1.5)	3.9 (1.5)	87	.151
- no. of relationships	RRELA	2.0 (0.9)	2.1 (0.9)	87	.359
- tasks without aggregates	RATTR=1	57%	51%		
- completely new requests	RSTAN=1	43%	47%		
- perceived request clarity	RCLAR	4.2 (0.8)	3.9 (0.9)	87	.093
LANGUAGE POWER MEASURES					
- no. of necessary queries	RQUER	4.4 (4.7)	3.2 (3.0)	87	.151
- tasks completely solvable	RSTRA>3	74%	84%		

Figures in parentheses give the standard deviation.

Table 4.3

Task Complexity and Language Power Measures

Due to limitations in the available data or to restrictions in the treatment, not all tasks could be answered; 15.6% for SQL tasks compared with 26.2% NLS tasks. On this basis it is concluded that SQL is somewhat more powerful functionally than NLS. It should be noted, however, that about three quarters of all tasks were answerable in both languages and all tasks were partially answerable.

An analysis of tasks showed that, although more queries, on the average, were necessary to solve a request in NLS than in SQL, due to the large variance in the data, the differences are not significant.

4.2.2 Effort Spent In Task Solution -

Effort was measured on the task level by 1) the number of queries submitted by subjects, and 2) the time subjects invested in working on a task. NLS subjects submitted approximately 50% more queries per task (and per session) than did SQL subjects (p<.001, n=138). However, the difference in total time working on a task was only about 15% (SQL - 108 minutes per task, NLS - 120). The difference in the average amount of time working on a task, between treatments, is not as great as would be expected based on the difference in the average number of queries submitted, probably because SQL queries require significantly more typing effort than does NLS (see query level analysis in section 4.3.1 below) and because data base searches are common to both treatments.

The number of queries per session and hence, per task, was lower during the second phase of the experiment than in the first phase, but only the drop in SQL is significant (see Table 4.4).

	Phase 1	Phase 2	Average	n	p (t-test)
Average Number Queries Per Session					
- NLS	10.4	9.9	10.1	65	.773
	(8.2)	(5.8)	(7.1)		
- SQL	7.4	4.5	6.2	73	.006
	(4.8)	(3.1)	(4.4)		
Average Number Queries Per Task					
- NLS	16.8	14.6	15.6		
- SQL	11.8	6.8	10.0		

Figures in parentheses show standard deviation.

Table 4.4

Average Number of Queries Submitted per Experiment Phase

Figure 4.1 shows the distribution of number of necessary queries per task (as judged by the experimenters - see section 3.3 and Appendix A), and of the number of queries submitted per session.

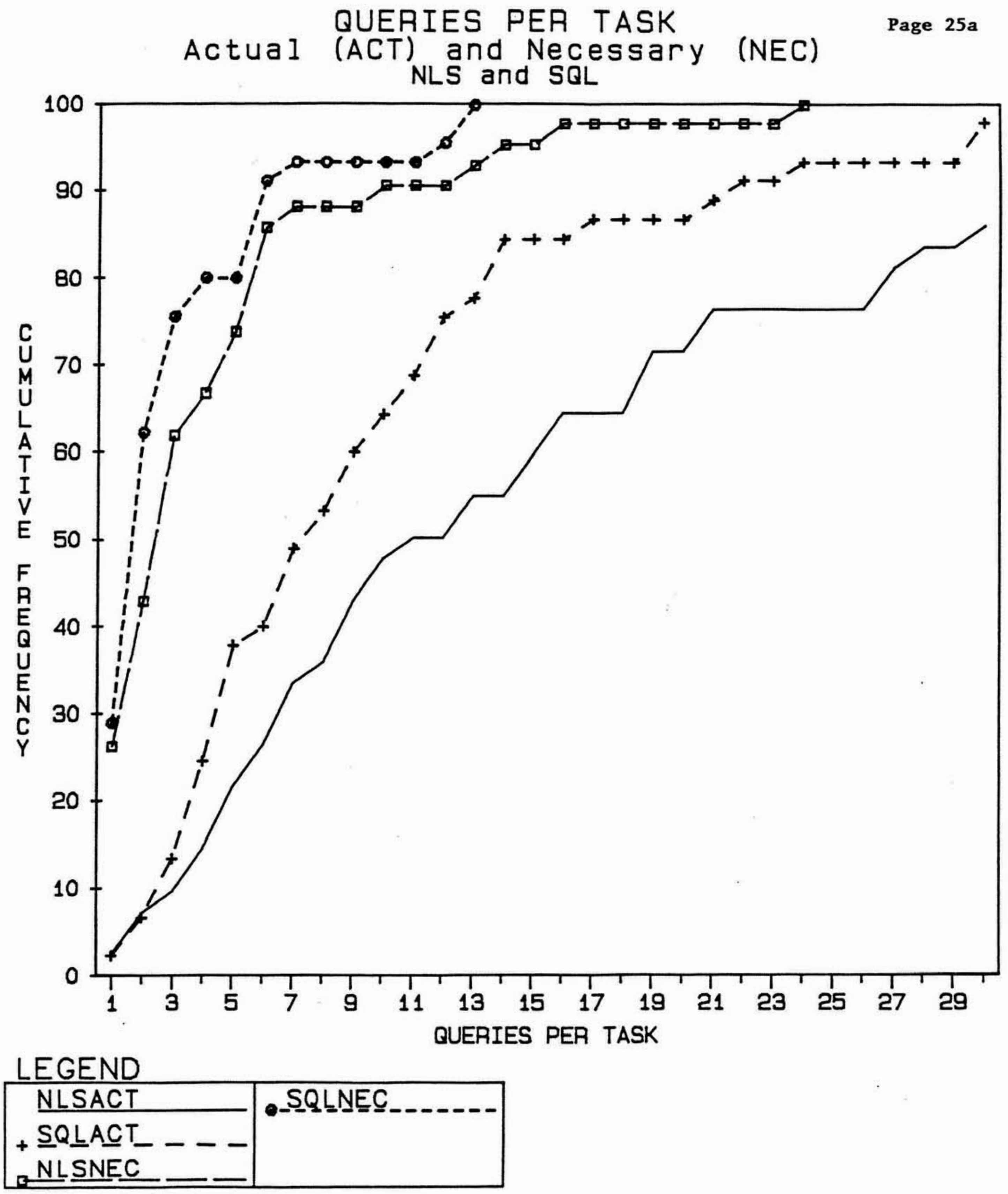

Figure 4.1: Task Solution Effort - Distributions of the Number of Actual and Necessary Queries Required to Solve a Task

----

Insert Figure 4.1 About Here

4.2.3 Task Performance -

The main performance measure at the task level is the proportion of the total number of tasks attempted that result in essentially correct solutions. By this measure, averaged over both phases of the experiment, SQL subjects were more than twice as successful in accomplishing their tasks as were NLS subjects (44.2% vs. 17.1% essentially correct solutions, see Table 4.5). This proportion hardly changes when only tasks that are fully solvable are taken as the basis of comparison (52.4% versus 23.6%). This implies that the difference in performance between NLS and SQL cannot be attributed to limited functionality in NLS alone.

Task Outcome	Phase 1		Phase 2		Average	
(RPATH)	NLS	SQL	NLS	SQL	NLS	SQL
essentially correct	4.8%	39.1%	30.0%	50.0%	17.1%	44.2%
solved partially	47.5%	21.7%	20.0%	25.0%	34.2%	23.3%
not solved	47.7%	39.1%	50.0%	25.0%	48.7%	32.5%

Table 4.5

Task Performance by Experiment Phase

The results for SQL do not show significant differences over time or among subjects. However, it should be remembered that the number of queries per task decreased during the second phase suggesting an improvement in performance (Table 4.4). On the other hand, for NLS, the success rate improved from 4.8% in the first phase to 30.0% in the second phase, without a significant change in the number of queries per task. There were strong individual differences between subjects using NLS, with one accounting for more than half of the successful task completions (while only 12.5% would be expected based on a random distribution of completions). Part of the poor performance of NLS during the first phase of the experiment was related to several language system bugs that were not fixed until the phase was almost complete (see discussion in section 5).

An analysis of the 28 paired tasks was performed to determine whether task performance changed when differences in task content were controlled (Table 4.6). The results demonstrate once more that SQL subjects performed better than NLS subjects in the first phase, but that the difference was much smaller in the second phase. It is interesting that for several of the tasks, the performance of natural language subjects was actually superior to that of the more structured language. No special characteristics of these tasks could be determined, except that they typically were solved by a number of independent, easy natural language queries, suggesting that the ease with which a task can be decomposed may be a factor in identifying situations where natural language may have an advantage.

	Requests For Which			
	NLS Better	SQL Better	Both Equal	Total Paired
phase 1	1 (8.3%)	9 (75.0%)	2 (16.7%)	12
phase 2	4 (25.0%)	8 (50.0%)	4 (25.0%)	16
total	5 (17.9%)	17 (60.7%)	6 (21.4%)	28

Table 4.6

Task Performance - Paired Analysis

A nonparametric correlation shows that success (RPATH) is negatively associated with the degree of novelty of a task (RSTAN: tau=-.200, p=.014, n=87) and the number of entities involved (RENETT: tau=-.158, p=.033, n=87). This finding is to be expected as both the degree of novelty of a task and the number of entities involved in a task are measures of task complexity.

The minimum number of necessary queries per task (RQUER), another form of difficulty measure, shows a negative relationship with success (RPATH) for both treatments (table 4.7). However, as table 4.8 suggests, a 'U' shaped relationship is found with the actual number of queries submitted (SRQUER). This can be explained as follows. Simple tasks need a relatively low number of queries to answer, accounting for the improved probability of success at low values. Then, a relatively large number of queries per task does not necessarily mean that the task was difficult; it may be that mistakes of one type or another caused a subject to use a number of unnecessary queries in answering a relatively simple task. In other words, it is reasonable that the probability of answering a request should decrease (up to a point) with the number of queries submitted and then increase again.

RQUER no. necessary queries	SQL success	NLS success
1	58.4%	20.0%
2-4	39.1%	17.5%
5 and more	37.5%	14.2%
average	44.2%	17.1%

success is defined as RPATH=essentially successful

Table 4.7

Task Success by the Number of Necessary Queries

SRQUER no. submitted queries	SQL success	NLS success
1 - 5	30.8%	13.3%
6 -10	20.0%	8.0%
more than 10	36.4%	12.5%
average	28.6%	12.5%

success is defined as RPATH=essentially correct

Table 4.8

Task Success by the Number of Actual Queries per Session

The consistency of these findings, that the same 2-3 to 1 difference in performance between subjects using NLS and those using SQL appears at all levels of analysis (figure 4.2), using a variety of measures and independent of task complexity, suggests a possible systematic cause. With this in mind, an analysis of the problems preventing success in each treatment was performed.

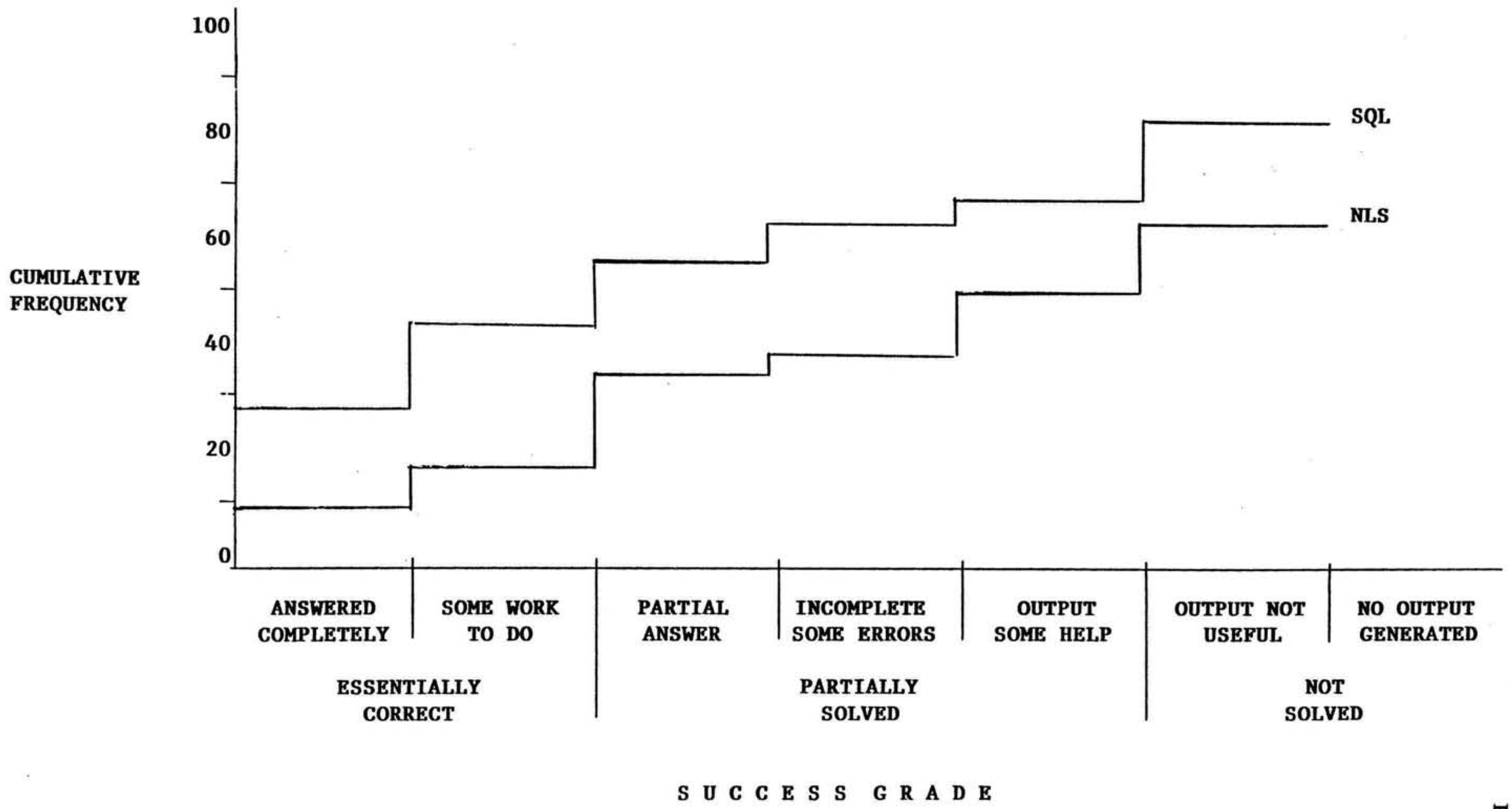

Figure 4.2: Comparison of Task Solution Performance

----

insert figure 4.2 about here

4.2.4 Major Problems Preventing Success -

One important reason why 59.4% of all NLS sessions and 42.3% of all SQL sessions did not yield useful output was technical problems with the user interface (SQL 21.1%, NLS 25.0% of all sessions). Interestingly, however, the specific problems encountered are different from language to language, and between the session and task level (Table 4.9). Apparently, SQL subjects were able to overcome interface problems in later sessions while NLS advisors were not. Only 7.0% of all SQL tasks had interface problems listed as the main reason for failure as compared to 22.0% of all NLS tasks, even though the operational environment was identical.

Main Problem	Task Level		Session Level	
(QPSOURCE)	NLS	SQL	NLS	SQL
no problem	14.6%	37.2%	9.4%	23.9%
not answerable (data)		7.0%		8.5%
lack of functionality	24.4%	2.3%	31.3%	1.4%
user problem	9.8%	34.9%	10.9%	42.3%
interface problem	22.0%	7.0%	25.0%	21.1%
system unavailable	7.3%		9.4%	2.8%
combination of problems	22.0%	11.6%	14.1%	

Table 4.9

Main Reasons for Task and Session Failures

Each treatment also exhibited different patterns of failure. In SQL, the most frequent cause of failure (34.9% of all tasks, 42.3% of all sessions) were user errors such as typos, syntax errors, or semantically inadequate queries. In contrast, the main problem in NLS was a lack of functionality in language or application design (24.4% of all tasks, 31.3% of all sessions). Furthermore, NLS subjects encountered system unavailability or a combination of several different problems substantially more often than SQL subjects.

4.2.5 User Perceptions -

Subjects were generally able to evaluate their own success realistically, both predicting it before a session and evaluating it afterwards, although the earlier expectations tended to be somewhat more optimistic (Table 4.10). Subjects also tended to underestimate their actual performance. There was a clear preference for the suitability of SQL over NLS (n=138, p<.001) for the tasks performed.

Evaluation Criterion (session)	Code	NLS	SQL
at least partially successful	RPATH<4	26.6%	45.1%
rather sure about strategy	SRSTRA>3	21.5%	39.4%
reported as partially successful	SRUSPR	17.3%	39.1%
average suitability grade	SRUSLS	2.3 (0.9)	3.3 (1.1)

Figures in parentheses show standard deviation

Table 4.10

Actual and Perceived Success

As might be expected, a subject's evaluation of language suitability (RUSLS) of a session is highly correlated with the actual success (RPATH, tau=.334, p=.001, n=138). On the other hand, the (ex-ante) correlation between clarity about the solution strategy (RSTRA) and actual success (RPATH) is not significant (tau=.079, p=.134, n=138). As table 4.11 shows, the evaluation of language suitability in relation to actual performance is similar between languages: suitability codes 3 and 4 exhibit actual result distributions, between languages, that are quite comparable.

Suitability	Actual Result					
	NLS			SQL		
Grade	correct	partial	not	correct	partial	not
1	0.0%	22.2%	77.8%			
2	8.8%	29.3%	61.8%			
3	20.0%	30.0%	50.0%	17.4%	30.4%	51.2%
4	28.6%	57.1%	14.3%	27.8%	44.4%	27.8%
5				72.7%	18.2%	9.1%

Table 4.11

Actual Success and Perceived Language Suitability

Subjects tended to be consistent in evaluating the results of a session. Perceived request clarity is found to be positively associated with the clarity of the solution strategy (SRCLAR - SRSTRA, tau=.455, p=.001, n=138) and the perceived language suitability (SRCLAR - SRUSLS, tau=.156, p=.020, n=138), but negatively associated with the perceived complexity of the task (SRCLAR - SRCOMP, tau=-.339, p=.001, n=138).

As shown in Table 4.12, the most frequent reasons subjects gave for failure to be successful in a session were lack of time (SQL 31.9% of all sessions, NLS 20.7%), and "couldn't figure out how to do it" (SQL 20.3%, NLS 51.7%). This is consistent with subjects' differences in perceptions about the suitability of the two languages.

SRUSPR Code Value	NLS	SQL
Successful Sessions		
- request completed	5.2%	34.8%
- partial output	12.1%	4.3%
Unsuccessful Sessions		
- not enough time	20.7%	31.9%
- couldn't figure out...	51.7%	20.3%
- system unavailable	3.4%	1.4%
- other reasons	6.9%	7.2%

Table 4.12

Perceptions of Success and Reasons for Failure

4.3 Query Level Results

4.3.1 Effort Spent In Query Generation -

Most subjects decomposed a task into sub-tasks. Only 15% of the queries attempted to answer complete requests and there were no differences between treatments in this regard.

SQL subjects used about three times the number of tokens per query as did natural language subjects (Table 4.13). If the assumption is made that subjects are attempting to accomplish a sub-task of the same difficulty in both treatments, then SQL appears to be more verbose than NLS.

	Phase 1	s.d.	Phase 2	s.d.	Total
SQL	34.15	20.08	34.27	21.40	34.19 (321%)
NLS	9.09	4.0	12.35	5.9	10.64

Table 4.13

Mean Query Length (QLGT)

Support for the efficiency of NLS in expressing a query also comes from the task level analysis. Based on the average number of queries per session and the average duration of a session, SQL subjects used about 40% more time, on the average, to formulate and execute a query than did NLS subjects (10.8 minutes in SQL vs. 7.7 minutes in NLS). However, as shown in Table 4.14, the shorter time to generate and execute a query in NLS was offset by the 58% larger number of queries used per task. Never the less, NLS retained an advantage in input length, even at the task level.

	Phase 1	s.d.	Phase 2	s.d.	Total
SQL	5.74	4.4	3.77	2.7	5.10
NLS	8.88	8.5	7.01	4.8	8.00 (58%)

Table 4.14

Mean Number of Queries per Task (Measured)

4.3.2 Success In Query Execution -

Subjects were about twice as likely to complete a query in SQL than in the NLS, yet completion occurred, at best, only in about a quarter of the cases -- much less than would be expected based on laboratory results (Table 4.15).

	Phase 1	Phase 2	Mean	s.d.
SQL	24.7%	27.6%	26.5%	2.05%
NLS	16.6%	13.7%	15.2%	2.05%

Table 4.15

Mean Rate of Query Completion (QPATH)

SQL subjects were also about twice as likely to have no error in their query as NLS subjects and were about two to three times more likely to get correct or partially correct output in SQL than they were in NLS (Table 4.16). These results are again about one quarter of what would be expected on the basis of the results of previous laboratory experiments.

	Phase 1	Phase 2	Mean	s.d.
Essentially Correct Output				
SQL	12.7%	14.9%	13.8%	1.56%
NLS	5.0%	2.6%	3.8%	1.70%
At Least Partially Correct Output				
SQL	17.9%	24.6%	21.25%	4.74%
NLS	7.9%	8.3%	8.10%	0.28%

Table 4.16

Probability of Obtaining Correct Output (QOQUAL)

If the same grading scheme is used as in the laboratory experiment (Welty category scores), the results become quite similar to those found in the laboratory experiments. It is evident that the results depend on the method of scoring, especially on what one assumes to be "correctable" by a "good" natural language system.

	Phase 1	Phase 2	Total
Essentially Correct Queries			
SQL	47.1%	44.0%	46.0%
NLS	24.2%	20.4%	22.4%
At Least Correctable Queries			
SQL	55.3%	60.4%	57.0%
NLS	70.0%	81.5%	75.5%

Table 4.17

Query Quality as Measured by Category Scores (QGRADE)

4.3.3 Errors Encountered And Sources -

Errors and their sources were classified as follows:

Interface. Line dropped, noise, terminal problem, communications switch problem, etc.

OS. Insufficient CPU cycles, insufficient amount of main memory, time out, cancel by system operator, etc.

Subject Typing. Typing error made by subjects, misspelling, etc.

Subject Language. Syntax error or semantic error made by subjects. In NLS, it might be failure to use a grammatically correct or complete sentence.

Language Logical. Language bug or feature that did not work.

Application Design. Specification error in designing the application, e.g., a word not defined in the application lexicon.

As shown in table 4.18, the patterns of error categories are quite different between treatments (as they were at the task level). In NLS during phase 1, the three greatest contributors were system, application design, and subject language errors. During phase 2, they were subject language, application design, and NLS logical. In other words, NLS subjects were having difficulty in getting their input query accepted (parsed) by the language system. SQL subjects, during phase 1, had subject typing, system, and subject language as the three greatest contributors to errors. During phase 2 they were subject typing, language logical, and system errors. It appears that SQL subjects were having difficulty correctly entering their queries.

	Phase 1	Phase 2
NLS		
No Error	11.1%	10.5%
Interface	10.5%	9.3%
System	23.0%	8.0%
Typing	9.9%	11.2%
Language Use	15.2%	25.1%
NLS Logical	9.6%	11.2%
Application	15.7%	19.8%
Other	3.8%	4.8%
SQL		
No Error	23.7%	24.6%
Interface	7.6%	1.5%
System	17.5%	11.1%
Typing	30.6%	31.3%
Language Use	14.1%	19.4%
SQL Logical	1.0%	-
Application	.3%	-
Other	5.2%	12.1%

Table 4.18

Query Level Error Categories (PSOURCE)

4.3.4 Error Recovery Strategies -

Two different error recovery strategies were indicated in the data. Given that a query had failed, natural language subjects were almost two times as likely to rephrase a query than were SQL subjects (Table 4.19). SQL subjects, on the other hand, were about two and one half times as likely to attempt the same query again as were natural language subjects.

	Phase 1	Phase 2	Total
Attempt the Same Query Again			
SQL	30.2%	29.1%	30.0%
NLS	12.0%	10.9%	11.4%
Rephrase Query			
SQL	22.0%	15.7%	20.0%
NLS	34.1%	40.6%	37.2%

Table 4.19

Error Recovery Strategies

4.3.5 External Factors Influencing Performance -

One of the factors that negatively influenced performance was that subjects were connected to the computer system remotely using 300 Baud dial up telephone lines instead of being hard wired (subjects also used printing terminals instead of CRTs). In addition, these communication lines were noisy. It was evident that this operating environment was the cause of a number of problems. In an attempt to simulate subjects' performance in a better environment, all queries that had error causes relating to interface or communication were removed from the population. When statistics were recalculated (reducing the number of queries by 13 - 35%), a 25-30% improvement in partially correct output resulted (Table 4.20)..

	Partially Correct Output Phase 1	Phase 2
Actual		
NLS	7.9%	8.3%
SQL	17.9%	24,6%
Noise Queries Removed		
NLS	10.8%	10.1%
SQL	23.6%	28.4%

Table 4.20

Comparison of Performance with Noise Queries Removed (QOQUAL)

4.4 Phase 3 Experiment

In phase 3 of the Field Experiment subjects were given a choice of treatments. It was reasoned that if subjects had the opportunity to select a treatment, assuming they were equally familar with both treatments, they would select the one that, 1) involved the least effort, that is, was best suited to the task, and 2) was most likely to produce correct output.

4.4.1 Method -

Five of the eight subjects participated in phase 3 because it took place several weeks after the completion of the prior phase. Subjects were given a practice session in which they worked on answering requests using both languages.

It was believed that certain forms of a task would be easier to solve in one language than the other. The requests from phase 1 and 2 were reviewed and, based on prior success, six forms (patterns) of tasks were selected: two where NLS was clearly superior, two where

SQL was clearly superior, and two where the languages were equal. Six new tasks were created based on these requests.

Subjects were given the new tasks and told to use the most appropriate language in accomplishing it. Subjects were to spend about one hour on each task.

4.4.2 Results -

Table 4.21 presents the results of phase three. It is evident that SQL was selected for most of the tasks. Twenty six tasks were attempted using SQL, while only 4 were attempted in NLS. It is also evident that the bias in the task (for a particular treatment) was not a factor; subjects appear no more prone to selecting the treatment toward which a task was biased.

		Treatment Selected	
		NLS	SQL
Task Bias			
NLS	- 1	1	4
	- 2	1	4*
SQL	- 1	0	5
	- 2	1	4
none	- 1	0	5*
	- 2	1	4
		4	26

* - a subject attempted to exit SQL and enter NLS, but was unsuccessful.

Table 4.21

Treatment Preference of Subjects

4.5 Questionnaire

Subjects were given a questionnaire at the completion of phase 2 asking them which language they, in general, prefered to work with and their reasons for this selection. SQL was prefered by all 8 subjects. The major points made were the following:

1. SQL was predictable; it seemed to do things in a consistent manner. It was not obvious what NLS would do. A minor rephrase of a correct NLS query might not work.

2. When NLS errors occurred, it was not clear what had caused the error or what action should be taken.

3. SQL constructs were difficult and error prone for complex queries. They were often easier to formulate in NLS.

4. Limitations in formatting and sorting in NLS detracted from its suitability.

4.6 Video Taping

Each subject was video taped for approximately one hour working on a task. Preliminary content analysis of this data showed results similar to the questionnaire data; a high degree of frustration when using NLS.

5.0 INTERPRETATION

The results of the two primary levels of analysis will be briefly summarized, followed by a discussion of the meaning of these results.

5.1 Task Level Results

SQL outperformed NLS, in terms of essentially correct task completion, by more than 2:1 (SQL - 44.2%, NLS - 17.1%) averaged across both phases of the experiment. The poor performance of NLS during the first phase (due to bugs in the prototype system, a lack of functionality, and the early stage of the application system) makes these results somewhat misleading. A more realistic ratio would be the 5:3 of the second phase (SQL - 50.0%, NLS - 30.0%). If the criterion is relaxed somewhat to include tasks that were partially solved, the differences between treatments lessen (Table 5.1). SQL was superior to NLS in 61% of the paired tasks, while NLS was superior in only 18%, or a performance difference of about 3:1.

	essentially correct or partially solved tasks		
	phase 1	phase 2	average
NLS	52.3%	50.0%	51.2%
SQL	60.8%	75.0%	67.9%

Table 5.1

Task Performance Summary

The performance of both SQL and NLS are disappointing at the task level. It is difficult to conceive of middle level executives having the patience to stick with a system where the probability of successful task accomplishment is at best 70%. These findings suggest there is considerable work still to be done in producing high performance question-answering systems for end users.

In terms of effort at the task level, NLS subjects used about 50% more queries per task than did SQL subjects (NLS - 15.6, SQL - 10.0) averaged across both phases. This is offset by NLS's efficiency at the query level (see below).

In a natural language system, the responsibility for errors shifts from the user to the system. This pattern clearly shows up in problem analysis. The major reasons for failure to complete a task were a lack of language functionality (24%) and interface problems (22%) for NLS, and subject errors in using the language (35%) for SQL. The inability to format or sort output probably contributed heavily to NLS' difficulties here. The order in which variables are placed on a page of output are predefined by the application design in NLS. This restricted a subject's ability to produce a custom report.

5.2 Query Level Results

SQL outperformed NLS, in terms of producing partially correct output from a query, by more than 3:1 (SQL - 21.3%, NLS - 8.1%) averaged across both phases. Although SQL appears superior to NLS, subjects using both languages performed much more poorly than had been expected.

Part of the difficulty in NLS, besides the prototype nature of the system, with its corresponding assortment of bugs and lack of functionality that have previously been mentioned, was the absence of almost any constructive feedback when an error occurred. It was usually impossible for subjects to determine what portion of the input had caused an error and what action should be taken to produce a correct query. This meant that subjects could not really debug NLS statements, suggesting a potential significant problem for natural language systems, especially bottom-up, syntax dependent ones.

In SQL, the complexity of the query structure introduced a number of errors. Subjects tended to omit a qualifier or not have the proper form of a relation or attribute name.

This difference explains why subjects tended to use different error recovery strategies in the two treatments. NLS subjects were about two times as likely to rephrase a query than were SQL subjects. Rephrasing is equivalent to constructing a new form of the query rather than sticking with the previous form and attempting to debug the query. SQL subjects, on the other hand, were about two and a half times more likely to attempt the same query again - that is, attempt to debug it.

In terms of effort expended for input, SQL queries averaged about three times the length of NLS queries (SQL - 34.2, NLS - 10.6 tokens per query) taken across both phases. Even if the 50% greater number of queries per task used by NLS is taken into account, NLS appears more concise than SQL. This finding, which was confirmed in the second laboratory experiment [22], holds real promise for natural language systems. When natural language configurations map efficiently into real world concepts (e.g., in the query, "List the Italian alumni") natural language can be extremely efficient.

In analyzing the problem sources, it is interesting to observe that the percentages of queries failing because of typing errors (SQL - 31%, NLS - 10%) are related in the same way as the length of queries (SQL - 34 av. tokens, NLS - 11 av. tokens): SQL and NLS show approximately the same low rate (1%) of typing errors per input token. Another point is the large number of failures (18%) attributed to omissions in the application-specific language design. In contrast to the approach taken by Krause [8], the application-specific lexicon was not changed during the experiment. The results suggest that a long period of adaptation to users may be required for successful NLS operation. Finally, the query level results confirm the importance of a smoothly functioning operating environment that would have increased performance in <u>both</u> languages by at least 30%.

As with the results at the task level, it is hard to picture professionals using a system that, at best, has a 26% probability of accepting a query and producing correct output.

5.3 Results In Perspective

Prior to accepting these results, several alternate explanations must be considered. The first issue is whether differences found in parameters between treatments are significant. In other words, are these differences meaningful? In general, at both the task and the query levels, subjects using SQL outperformed subjects using NLS, by meaningful differences, using a variety of measures.

The next issue is whether these differences represent real differences between treatments, or whether there is evidence that they might have been caused by other factors, such as individual differences among subjects. The research strategy used was twofold: to select subjects on the basis of their similarity, and to use a balanced design. An examination of the grand means and standard deviations for the major statistics indicates, with one exception, consistency between treatment groups (Table 5.2). This suggests that differences between treatments represent actual differences.

	Grand Mean	s.d.	p	Delta
Task Level				
Performance (essentially correct)				
- NLS	17.4	17.8		25.2
- SQL	44.5	7.7		10.9
Query Level				
Number of Tokens per Query				
- NLS	11.2	3.0		3.3
- SQL	34.2	0.1	<.001	0.1
Rate of Completion				
- NLS	15.2	2.1		-2.9
- SQL	26.5	2.1	<.001	2.9
Output (essentially correct)				
- NLS	3.8	1.7		-2.4
- SQL	13.8	1.6	<.001	2.2

Delta is the difference between the phase 2 and the phase 1 value of the parameter. A large value of delta compared to the Grand Mean indicates an order effect.
Grand Mean is the mean of the parameter across both phases.
p is the probability that the grand mean values of the parameter for the two treatments are drawn from the same population - t-test.

Table 5.2

Between Phase and Order Effect Statistics

This is not to say that there were no differences between subjects in using a treatment. Quite the contrary: particularly with NLS, the variance is quite high. For example, successful solutions came from only three of the eight subjects, and one subject accounted for more than half of the successful tasks. This suggests that one either understands how to use NLS, or not. There is little middle ground. The variance in SQL is less extreme, suggesting a language with more broad appeal. However, taken on the average, over some number of subjects, the differences in parameters do not appear attributable to differences in the composition of treatment groups.

The third issue is whether an order effect exists. The presence of an order effect between the two phases of the field experiment would tend to indicate that, either, 1) subjects' performance was changing as a function of time, or 2) some factor influencing the experiment was changing over time. A change in subjects' performance might be attributable to continued learning, possibly as a result of inadequate initial training. Factors changing over time that might influence the experiment include the quality of the languages and the tasks subjects were given.

The findings are mixed concerning an order effect. At the task level, the data suggests that task complexity decreased during the second phase of the experiment. Whether this was because principals became discouraged by the lack of prior task solutions or they ran out of their normal work is not clear. However, the slight decrease in task complexity during the second phase is probably accountable for part of the improvement in performance observed at the task level.

At the query level, the data between phases is quite consistent and the differences are considerably smaller than the differences between treatments suggesting no order effect (Table 5.2). In general, the difference between phases (delta) is less than 50% of the difference between treatment grand means.

Finally, are the results consistent over a variety of different methods of testing and gathering data? That is, do the results evidence convergent validity? Here it is noted that the results of both phases of the field experiment, the third phase in which subjects selected their prefered treatment, and the results of a questionnaire all indicate the same outcome.

5.4 Hypotheses

The hypotheses developed in section 2.7 are now considered in light of the findings of the field experiment. H1, that there would be no difference in performance between treatment groups, is rejected. Almost all tests at both levels of analysis showed SQL to be superior to NLS. It still has to be explained why the results of the field experiment departed so radically from those of the laboratory experiment.

H2, that subjects using NLS will be more efficient than subjects using SQL, is accepted conditionally, in terms of input token length (it is unclear in terms of time). Certainly the potential advantage in efficiency of NLS should be evident when proper feedback is provided.

H3, that the performance of subjects will be negatively related to task difficulty, is accepted. Although task difficulty appears to be a multidimensional concept, most measures showed a negative relationship with performance.

The final hypothesis, that the performance of subjects will be negatively related to perceptions of task difficulty is accepted. The second part of the hypothesis, that performance will be positively related to a subject's perceived understanding of a solution strategy, could not be confirmed.

5.5 Qualifications

The results of this study are qualified by a number of factors. First, as mentioned earlier, the contrast in this study is between a particular natural language system, NLS, and SQL, a structured data base query language. If another natural language system were substituted for NLS, or for that matter, if another reference language were used, it is likely that some of the results would change. It is extremely difficult to identify those findings that are fundamental to natural language systems and those that are related to the particular incarnation tested. The same comments apply to the application system. A second application in another area might have produced different results.

The second issue is the prototype nature of NLS. The absence of feedback to subjects and the lack of functionality during phase one certainly biased the results against NLS. However, these limitations in NLS were actually present, and the fact that the experimental design was sensitive enough to detect these problems illustrates the strength of the methodology.

Third, the poor operating environment consisting of remote telephone lines, printing terminals, and a heavily loaded machine certainly contributed to some of the frustration evidenced by subjects and, to some extent, to their generally poor performance. Yet, this factor alone cannot explain the difference in performance between the two treatments.

5.6 Comparison With Laboratory Experiment Results

It has been observed that the results of the field experiment were much poorer than expected on the basis of related laboratory experiments. Table 5.3 compares the results of the field experiment with two laboratory studies [20], [22]. A direct comparison of results may be misleading because of differences in research methods, subjects, and objectives between the studies. However, the comparison uses the same grading scheme [23] for all of the experiments.

Experiment	Evaluation Criterion	NLS	SQL
Task Solution Performance			
Field	% solvable tasks	73.8%	84.4%
Field	% essentially correct tasks	17.1%	44.2%
- phase 1	% essentially correct tasks	4.8%	39.1%
- phase 2	% essentially correct tasks	30.0%	50.0%
Actual Query Answering Performance			
Lab I	% essentially correct queries	71.1%	67.3%
Lab II	% essentially correct queries	44.6%	53.3%
Field	% essentially correct queries	22.3%	45.6%
Potential Query Answering Performance			
Lab I	% correctable queries	78.8%	76.9%
Lab II	% correctable queries	59.2%	68.6%
Field	% correctable queries	75.5%	57.0%

Table 5.3

Comparison of Field and Laboratory Experiment Results

Part of the discrepancy in scores between subjects in the laboratory and field experiments has to do with the greater complexity of the field setting (refer to the end of section 3.3). Scoring schemes used in laboratory settings cannot accurately capture this degree of complexity, and so are likely to overstate actual performance.

6.0 IMPLICATIONS

The implications of this study for both natural language question-answering systems design and for evaluation research are now considered.

6.1 Natural Language Systems

No superiority of natural language systems over formal languages could be demonstrated in terms of either query correctness or task solution performance. This is in agreement with previous studies of natural language concepts [15], [17] and systems [8]. However, it has been established that natural language queries are more concise and require less formulation time, thereby demonstrating potential advantages of an improved natural language system. The following design considerations have proven crucial.

Probably the most important observation is the importance of feedback in subject performance. Without proper error messages an operator is unable to debug an incorrect query, greatly reducing the problem solving strategies available. The quality of this feedback cannot be separated from the syntax and semantics of the language, if a complete view of performance is desired. Mechanisms for correcting trivial errors (e.g., spelling) without retyping whole queries must be provided.

The second observation is the importance of the total operating environment on the performance of subjects. Too often a relatively narrow or idealized view is taken in system evaluation which is misleading. Systems loading (and by implication, system performance), the communications interface, layered systems all have the potential of drastically reducing actual subject performance. While given resources and time, any of these factors can be removed, they frequently are all present in real life settings. More field evaluations are needed to clarify the interplay between realistic operational settings and specific data base query languages.

Third, natural language application design differs from normal application system design. It is usually presumed in the Life Cycle approach to application design, that closure will be reached at the completion of testing. Although normal application systems evolve, this usually occurs in discrete steps or versions. With natural language application design, the process is much more iterative - queries that don't parse or that produce incorrect output serve as the source of changes. In this sense it is somewhat like using prototypes in the process of design to obtain user feedback. Of course, the difficulty is that this process may not be convergent and closure may never be reached. Also, once a user realizes that a feature does not work they will rarely attempt to use that feature again even if it has been fixed.

Finally, restricted natural language systems require training. While one of the advantages of natural language is purported to be the absence of training, our experience suggests otherwise.

None of these recommendations goes beyond the concept of the type of natural language system tested here. Thus, although the practical performance of the tested prototype was unsatisfactory, the underlying philosophy of the system cannot be rejected based on this study. It should be noted that most systems claiming to overcome these problems in a conceptually more elegant way (e.g., using knowledge-based AI techniques) are not even in a state where a study of this type could have been performed. The most popular commercially available natural language query system, Intellect [1], [5], follows a similar philosophy as NLS.

6.2 Field Research

The most important finding, in terms of research methodology, is that a research design can be created that is capable of detecting difference in performance between subjects using computer languages in complex work settings. The consistency of results between phases is encouraging for future efforts.

The next observation is the need to conceive a strategy that provides as much control as possible in the field setting. In this study, both controls for individual differences among treatment groups and the comparative nature of the study permitted dealing with a number of unexpected problems without compromising the study. Careful research designs do work and they are well worth the extra effort. However, care must be taken not to confound the results by selecting a work environment that is too 'realistic'.

Third, it is important to use multiple strategies in gathering data to improve the validity of the study. A concept (e.g., suitability of a language) should be tapped with as many different measurement methods as possible.

Fourth, it was not obvious how important the coding scheme would be or how difficult to develop. At the beginning of the study it was not clear what aspects of, for example, a query would be important. The types of problems that would be encountered were not known in advance, requiring that sample data be gathered and analyzed qualitatively <u>before</u> the coding scheme could be developed. There is a trade-off between the effort required to code the data and carrying a sufficient number of codes to capture parameters of interest. Skill in making this decision usually influences the success of the project.

Finally, it is difficult to separate theoretical issues from the details of a particular situation. For example, in attributing the causes of a finding to either the strategy used in implementing NLS, the specific materialization of the strategy, or the evaluation setting.

7.0 CONCLUSIONS

Returning to the three questions raised in the beginning of this paper. Given the specific natural language prototype evaluated, it does not appear that subjects with real work to do could use it more successfully than a formal query language. In the particular research setting, neither language permitted real work to be accomplished. How much of this related to the particular prototype version of NLS tested and how much is fundamental to NLS or to natural language systems in general, remains to be shown in further studies.

The results of this study do suggest that when real world concepts map easily to sufficiently qualified data base queries, natural language can have sigificant efficiency advantages over more structured languages. The challenge is determining how to build a resilient natural language system with more continuous and understandable operation.

Finally, there clearly is an interplay between subjects' problem solving behavior and the features of application languages they use. The inability of NLS to provide feedback on the cause of problems deprived subjects of the ability to debug queries, forcing them to adopt another problem solving strategy in order to accomplish their work. This rephrasing of a query to find a combination that would parse frequently became an end in itself drawing the subject away from the original task.

This research has shown that it is possible to study specific computer languages in real work settings. The methodology developed for this purpose did consistently measure differences in treatments over a variety of conditions.

The approach to experimental design reported in this paper should help to focus the attention of researchers on important questions that have to be answered as a prerequisite for the successful empirical study of query languages. The results of this evaluation concerning natural language challenge designers of newer, in particular, knowledge-based interfaces to offer empirical evidence of the practical superiority of these systems.

Appendix A - Task Level Measurement Scheme

The primary higher unit of analysis is the task level - the work product a subject is trying to accomplish. The general notion is that task performance will be a function of 1) the difficulty of the task, 2) the language used (treatment) to accomplish the task, and 3) the skill level of the subject.

A.1 Request Number

A request or task is identified by its Request Number (REQNO). The composition of a task was captured on a written form by subjects after their meeting with a principal. When more than one subject met with a principal, their separate task descriptions permitted a reconstruction of the original request. In general, few if any differences were found among subjects in task descriptions.

A.2 Complexity Measures

The specific measures consisted of:

RENETT - an objective, language independent measure of task complexity, based on the application entity-relationship model (figure 2.1), of the number of entities involved in a task.

RRELA - another objective, language independent measure of of task complexity, again based on the application entity-relationship model, of the number of relationships involved in a task.

It was expected that these two measures would be highly correlated.

RATTR - a subjective, language independent measure of the degree of difficulty of the operations required to accomplish a task.

Simple tasks, by this measure, might require just the print out of data from the file, while a more complex task might require that the data be sorted, aggregated or possibly grouped. (Certain language features were not operational during the study, for example sorting in NLS, requiring another measure, RSTRA, to indicate whether a task could be performed with a given language).

RQUER - a subjective, language dependent measure of the minimum number of queries required to accomplish a task.

RSTRA - a subjective, language dependent measure of the proportion of a task that can be solved in a given language.

RCOMP - a subjective, language dependent measure of the degree of difficulty in accomplishing a particular task.

RSTAN - an objective, language independent measure of how frequently a task had been requested - the uniqueness or novelty of a task.

RPUPR - a subjective, language independent measure of the purpose the principal had in mind in requesting the task.

RCLAR - is a subjective, language independent measure of how clear is the task that a principal wants performed.

SRQUER - an objective, language dependent measure of the actual number of queries used in a session.

Values for all of these codes were determined by the investigators.

A.3 Outcome Measures

RPATH - an objective, language independent, seven category measure, aggregated to three categories, that describes the outcome of a task. Categories are: essentially accomplished correctly, partially accomplished, or not accomplished at all (i.e., no output helpful in accomplishing the task was produced).

RPSRC - a subjective, language independent seven category measure identifying the major reason for not accomplishing a task.

STIME - an objective, language independent measure of the amount of time taken in a session.

A.4 Subject's Perceptions

During each session, subjects captured their perceptions about the tasks they were performing.

SRCLAR - a five category measure of how clear a task was to a subject.

SRSTRA - a five category measure of how certain a subject was about a solution strategy for a task.

SRCOMP - a five category measure of how complex a subject thought a task to be.

SRPURP - a measure of what a subject thought the purpose of a task was.

SRSTAN - a measure of how unique a task was for a subject.

SRUSPR - a measure of how successful a subject believed he was in accomplishing the task.

SRUSLS - a measure of the suitability of the language for the task.

Appendix B - Query Level Measurement Scheme

The following describe the major query level measures.

B.1 Query Number

A consecutive count of the number of queries in a session is given by QNO.

B.2 Query Description Measures

QLGT - an objective, language independent measure of number of tokens (words) in a query.

QRPR - a subjective, language independent measure of whether a query is an initial attempt to answer a subtask, or whether it is a rephrase of a prior query attempt within that session.

QOBJ - a subjective, language independent measure of whether a query attempts to solve the complete task or only a portion (subtask) of the task.

The following complexity codes were derived from the SQL representation of a query (remember that NLS maps to SQL for execution as both languages use the same underlying data base system). Thus, for NLS these complexity measures apply to the resulting data base query rather than to the input query as entered by a subject.

QVARS - an objective, language independent measure of the number of relation names in the 'FROM' clause, plus any relation names appearing in nested 'SELECT' clauses. This measure provides a count of the number of different concepts (entities) in a query.

QRESTR - an objective, language independent measure of the number of relational clauses of the form, 'field OP value', where 'OP' is a relational operator. This measure represents the complexity of the operations being performed on the entities of a query.

QJOINS - an objective, language independent measure of the number of 'JOINS' plus nested 'SELECT' clauses. This measure describes the complexity of the interaction among variables.

QTATTR - an objective, language independent, categorical variable indicating whether the 'SELECT' clause contains attributes only, data aggregates (e.g., "COUNT' only), or a combination of attributes and aggregates.

Since the syntactic components of the two languages are nOtot distinguished by these measures, they mainly represent different aspects of computational complexity. Formulation complexity is captured, primarily, by the descriptive measure, QLGT (token count).

B.3 Query Execution Description Measures

QPATH - a subjective, language independent, categorical measure describing the path taken by a query attempt (see figure 3.2).

QCHNG - a subjective, language independent, categorical measure describing the response of a subject after an error occurred.

QINTBY - a subjective, language independent, categorical measure of the agent that initiated the termination of processing.

B.4 Query Outcome Success Measures

QOQUAL - a subjective, language independent categorical measure of the quality of the output produced by the query, with reguard to the subtask.

QGRADE - a subjective, language independent, categorical measure of the quality of the query. It is essentially the same measure used by Welty and Stemple [23], and the same measure used to score the two laboratory experiments.

B.5 Query Outcome Problem Measures

QERR - a subjective, language dependent, categorical measure of the specific reason why a query failed to execute to completion. The codes were developed from a sample of logs for each language and tabulations of the codes provide an indication of the relative frequency of each category of failure.

QPSOURCE - a subjective, language independent, categorical measure of the source of the problem (QERR) that prevented a query from executing to completion. This measure is useful in removing certain categories of queries from the sample (for example, queries that had interface problems), and for control.

LPSRC - a subjective, language independent, categorical measure indicating the reason for an error. Categories are: subject, language, or application development.

REFERENCES

1. Artificial Intelligence Corporation (1982): "Intellect Query System Reference Manual".

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

3. Ford, W. R. (1981): "Natural-Language Processing by Computer - A New Approach," Ph.D. dissertation, The Johns Hopkins University, 81-15709, UMI.

4. Greenblatt, D., Waxman, J. (1978): "A Study of three database query languages," in Shneiderman, B. (ed.), Databases: Improving Usability and Responsiveness, Springer, New York, pp. 77-97.

5. Harris, L. (1978): "Experience with ROBOT in 12 Commercial Natural Language Data Base Query Applications," Proc. of NCC, pp. 365-368.

6. Jarke, M., Vassiliou, Y. (1982): "Choosing a Database Query Language", NYU Working Paper Series, submitted for publication.

7. Krause, J. (1980): "Natural Language Access to Information Systems - An Evaluation Study of its Acceptance by End Users," Information Systems 5, 4, pp. 297-319.

8. Krause, J. (1982): Mensch-Maschine-Interaktion in natuerlicher Sprache, Niemeyer Verlag, Tuebingen.

9. Lehmann, H. (1978): "Interpretation of Natural Language in an Information System," IBM?Journal of Research and Development 22, 5, pp. 560-571.

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

11. Miller, L. A. (1981): "Natural Language Programming: Styles, Strategies, and Contrasts," IBM Systems Journal 20, 2, pp. 184-215.

12. Petrick, S. R. (1976): "On Natural Language Based Computer Systems," IBM Journal of Research and Development 20, 4, pp. 314-325.

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

14. Reisner, P., R. F. Boyce, and D. D. Chamberlin (1975): "Human Factors Evaluation of Two Data Base Query Languages - Square and Sequel," Proceedings of NCC, pp. 447-452.

15. Shneiderman, B. (1980): Software Psychology, Little Brown and Co. Boston, MA.

16. Simmons, R. F. (1970): "Natural Language Question-Answering Systems: 1969," Communications of the ACM 13, 1, pp. 13-30.

17. Small, D.W., and Weldon, L.J. (1983): "An Experimental Comparison of Natural and Structured Query Languages," Human Factors 25, 3, pp. 253-263.

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

19. Tennant, H. (1979): "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.

20. Turner, J. A., M. Jarke, E. A. Stohr, Y. Vassiliou, and N. White (1984): "Using Restricted Natural Language for Data Retrieval: A Plan for Field Evaluation," in Y. Vassiliou (ed.), Human Factors and Interactive Computer Systems, ABLEX, Norwood, NJ.

21. Vassiliou, Y and M. Jarke (1984): "Query Languages: A Taxonomy," in Y. Vassiliou (ed.) Human Factors and Interactive Computer Systems, ABLEX, Norwood, NJ.

22. Vassiliou, Y, M. Jarke, E. A. Stohr, J. A. Turner, and N. White (1983): "Natural Language for Database Queries: A Laboratory Study," MIS Quarterly?7, 4, pp. 47-61.

23. Welty, C. and D. W. Stemple (1981): "Human Factors Comparison of a Procedural and a Nonprocedural Query Language," ACM Transactions on Database Systems 6, 4, pp. 626-49.