HUMAN FACTORS EVALUATION OF A TEXTUAL, GRAPHICAL, AND NATURAL LANGUAGE QUERY INTERFACES

by

John E. Bell and Lawrence A. Rowe

Memorandum No. UCB/ERL M90/12

22 February 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

HUMAN FACTORS EVALUATION OF A TEXTUAL, GRAPHICAL, AND NATURAL LANGUAGE QUERY INTERFACES

by

John E. Bell and Lawrence A. Rowe

Memorandum No. UCB/ERL M90/12

22 February 1990

HUMAN FACTORS EVALUATION OF A TEXTUAL, GRAPHICAL, AND NATURAL LANGUAGE QUERY INTERFACES

by

John E. Bell and Lawrence A. Rowe

Memorandum No. UCB/ERL M90/12

22 February 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

Human Factors Evaluation of a Textual, Graphical, and Natural Language Query Interfaces

John E. Bell
Lawrence A. Rowe

Computer Science Division-EECS
University of California
Berkeley, CA 94720

ABSTRACT

This paper describes a human factors experiment performed to compare three different interface styles for database query. Over sixty subjects with wide ranging computer experience performed queries of varying difficulty using either an artificial, graphical, or natural language interface. All three interfaces were commercial products. The experiment showed that none of the interfaces was best for all queries or users.

1. Introduction

Increasing computer power and decreasing costs is causing radical changes in user interfaces. Graphical and natural language user interfaces are now practical alternatives to alphanumeric user interfaces. This paper describes a human factors experiment that compared three different interface styles for database query: 1) artificial language (SQL), 2) graphical language, and 3) natural language.

Other researchers have compared different interfaces to query languages with varying success. Reisner, Boyce and Chamberlin compared a relational algebra query language (SQUARE) and a relational calculus query language (SQL) [Reis75]. Using 33 subjects, they compared these full function query languages and they found that the relational calculus was easier for non-programmers to learn.

Greenblatt and Waxman [GrWa78] and Boyle, Bury and Evey [BoBE83] performed experiments comparing SQL and QBE, each with 25 or fewer subjects. Greenblatt and Waxman concluded that QBE was better but they did not use actual systems. Boyle, Bury and Evey used actual systems and they found that subjects took longer to learn QBE and that SQL was better for some types of queries while QBE was better for others.

Shneiderman [Shne78] and Small and Weldon [SmWe83] performed experiments that compared SQL and natural language. Their results were inconclusive because the experiments were limited by experimental design problems (e.g., limited query language functionality and small subject sample size) and by the lack of a working natural language interface.

Jarke, et al. [Jark85] performed a field study that also compared SQL and natural language, but again with inconclusive results. SQL appeared better than natural language because users achieved a higher success rate on queries. However, there were very few subjects and little control of the experimental conditions.

Other experiments have been performed which studied only one interface (QBE [ThGo75] or NL [OgBr83], [FiSB85], [OgSo87], [Krau80], [Dame81]) or were designed to answer different questions (e.g., procedural versus non-procedural [WeSt81], different data models [Loch78], [BrSh78], or programming using natural language [Bier83]).

This paper describes an experiment comparing full function query languages with sixty subjects working on actual interfaces in a controlled environment. Subjects ranged in computer experience from none at all to experienced database users with programming experience. The tasks included simple queries (e.g., single table) to complex queries (e.g., multiple table, counting, and existential). Subject performance was analyzed across interfaces and across task types giving a rich and complete picture of the usability of these interfaces.

The remainder of this paper describes the experiment and the results. Section 2 describes the sample database and the three interfaces. Section 3 describes the details of the experiment, including its design, the subjects involved, and the treatment. Section 4 presents the experimental results, and section 5 contains our conclusions.

2. The Sample Database and Interfaces

The sample database included information about students, teachers, classes and activities for a high school. Figure 1 shows an entity-relationship diagram for the database. Entities are represented by boxes; relationships are represented by diamonds, and attributes are represented by circles.

The high school database was chosen because it would be familiar to all subjects and it is easy to understand. In addition, it has sufficient entities and relationships to allow a variety of simple and complex questions to be asked.

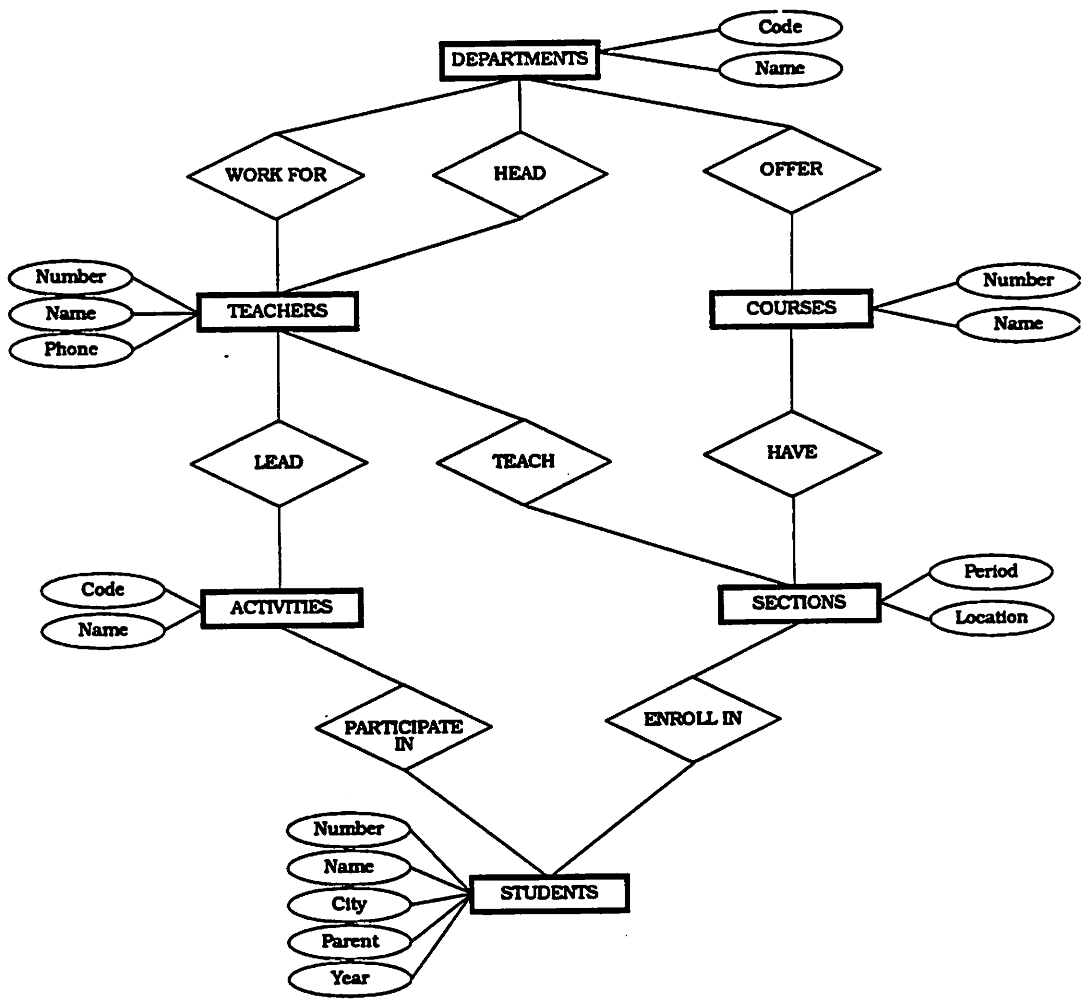


Figure 1. ER diagram of high school database

A fundamental problem encountered when comparing different interfaces for the same task is to find a fair method to present the task to the experimental subject. For example, a natural language description of the task is inappropriate because one of the interfaces being compared is a natural language interface which could bias the results. In addition, one task presentation was needed so that differences could be eliminated in how subjects using different interfaces understood the task.

A pictorial representation was chosen to present the task. Figure 2 shows an example.

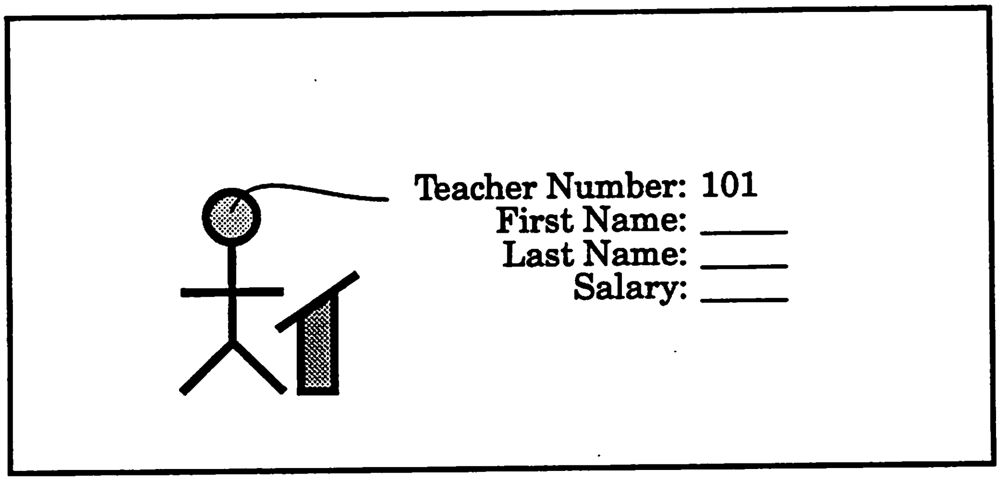


Figure 2. An example of pictorial task presentation.

The task represented in the figure is to find the first and last names and salary of teacher number 101.[1] The icon represents the entity, in this case a teacher, and the captions represent the attributes. Attributes with a value are restrictions and attributes without a value specific the information to be retrieved. Other icons used in the pictures are shown in Figure 3. Subjects could refer to a sheet showing these icons and their meaning during the experiment.

[1]The actual representation was slightly different but the difference did not affect understanding by the subjects. For more details, see [Bell 90].

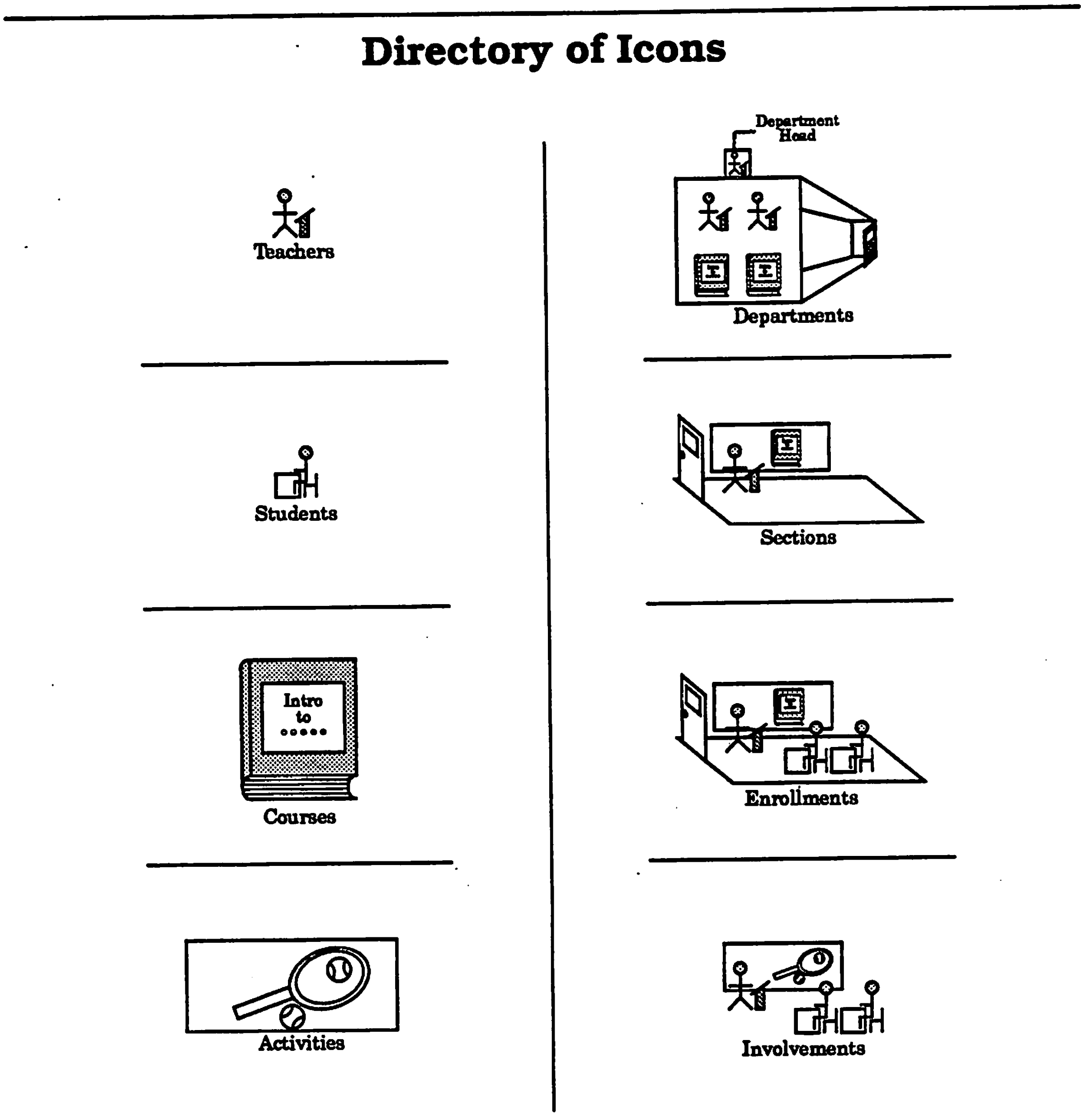


Figure 3: The icons used for pictures in the high school database.

Figure 4 shows a task description for a two table query. the task is to list all courses taught by each teacher. Notice that the join between teachers and sections is represented by using attributes in the different entities. The remainder of this section shows how this query can be entered into the three interfaces.

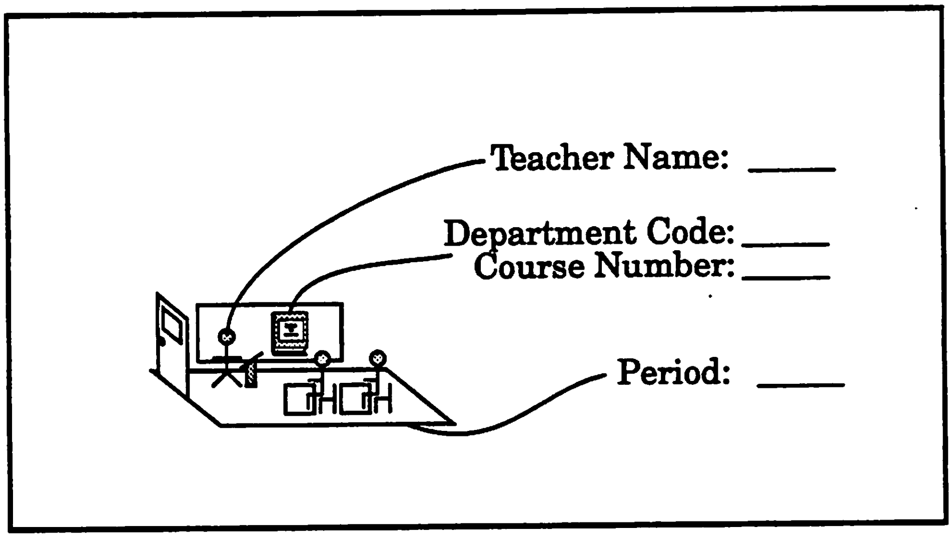


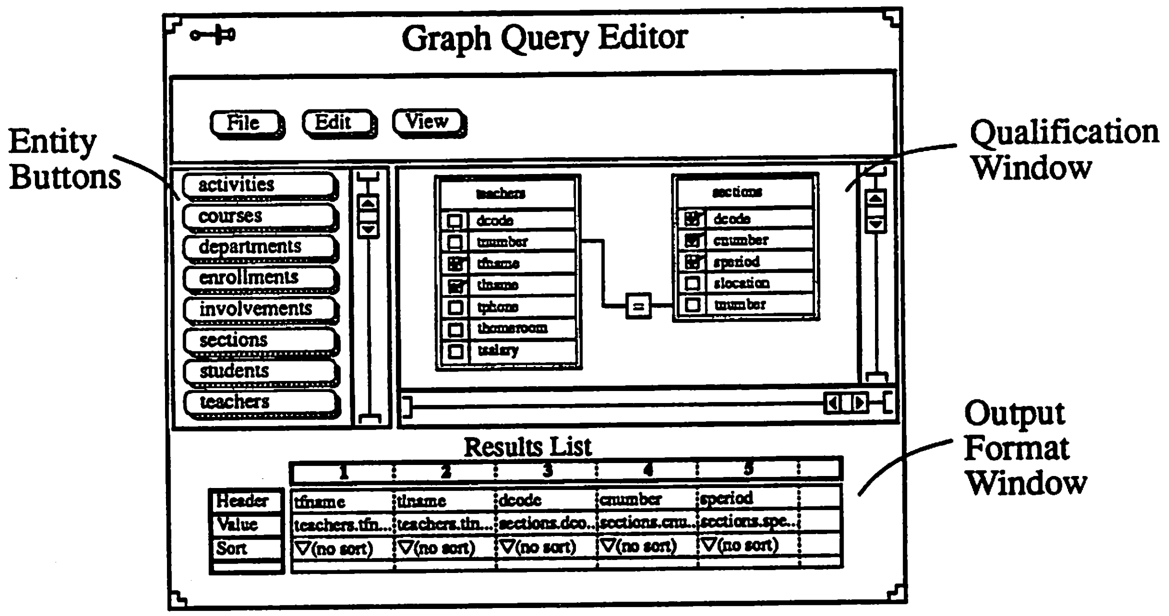
Figure 4: Task description: List all courses taught by each teacher.

The first interface was an artificial language interface that used SQL (AL). It uses English keywords (e.g., *select*, *from*, and *where*) to make it more readable and easier to remember. SQL is claimed to be user friendly because it is a nonprocedural language. An SQL query that solves the sample task is:

```
select teachers.tfname, teachers.tlname,
   sections.dcode,
   sections.cnumber, sections.speriod
from teachers, sections
where teachers.tnumber = sections.tnumber
```

The second interface was a graphical language, called Simplify developed by Sun Microsystems (GL). Earlier versions of Simplify were developed at Xerox PARC [Catt80]. Simplify is essentially a graphical interface to an SQL processor. A query constructed in Simplify is translated into an SQL query which is then run. Consequently, Simplify and SQL users are specifying the same commands but they enter the commands in very different ways.

Figure 5 shows a Simplify window that contains the query for the sample task.



	1	2	3	4	5
Header	tfname	tlname	dcode	cnumber	speriod
Value	teachers.tfn...	teachers.tln...	sections.dco...	sections.cnu...	sections.spe...
Sort	▽(no sort)	▽(no sort)	▽(no sort)	▽(no sort)	▽(no sort)

Figure 5: A sample Simplify screen.[2]

The query is created by the following steps:

- Click on entity buttons for the tables in the query (i.e., teachers and sections). The entity boxes in the qualification window are displayed when the entity is selected.
- Click on the attributes in the entity boxes that are to be included in the query. Selected attributes are indicated by check marks and are automatically entered into the output format window.

[2]The entity boxes in Figure 5 are shown side-by-side to enhance readability.

- Click on tnumber in teachers, select "Create a join" in a pop-up menu, and click on tnumber in sections. This specifies the join between teachers and sections. A line joining these two attributes is displayed.

The query is then executed by choosing "Execute query" in a pop-up menu. The query result is displayed in another window.

The third interface was a natural language interface, called DataTalker, developed by Natural Language, Inc. (NL). It allows users to specify a query by entering plain English on a keyboard.

One solution to the sample task is the following:

```
Show the courses taught by each teacher
```

It is important to realize that this is only one way among many that the user can enter to request this information. For example, another way to ask for this information is:

```
List the teachers and their courses
```

One objective of a natural language interface is to allow a variety of questions to access the same data.

All interfaces used the Ingres DBMS to store data. The SQL interface used in the experiment was the *isql* full-screen editor supplied with Ingres 5.0. The Simplify interface used was version 1.0 (beta) released in May 1989. The DataTalker interface used was version 3.0 released in March 1989. The experiment was performed on a Sun 3 computer. The same database was used for all subjects.

3. The Experiment

The experimental design is shown in Figure 6. Subjects were broken into five groups (the vertical axis), and were then randomly assigned to three treatments (the horizontal axis). Each subject worked through two phases: a learning phase and a performance phase. The learning phase was composed of task levels that covered seven types of queries. The query types taught in each level are described in Figure 7. The performance phase included queries to test how well the user learned to use each interface during the experiment.

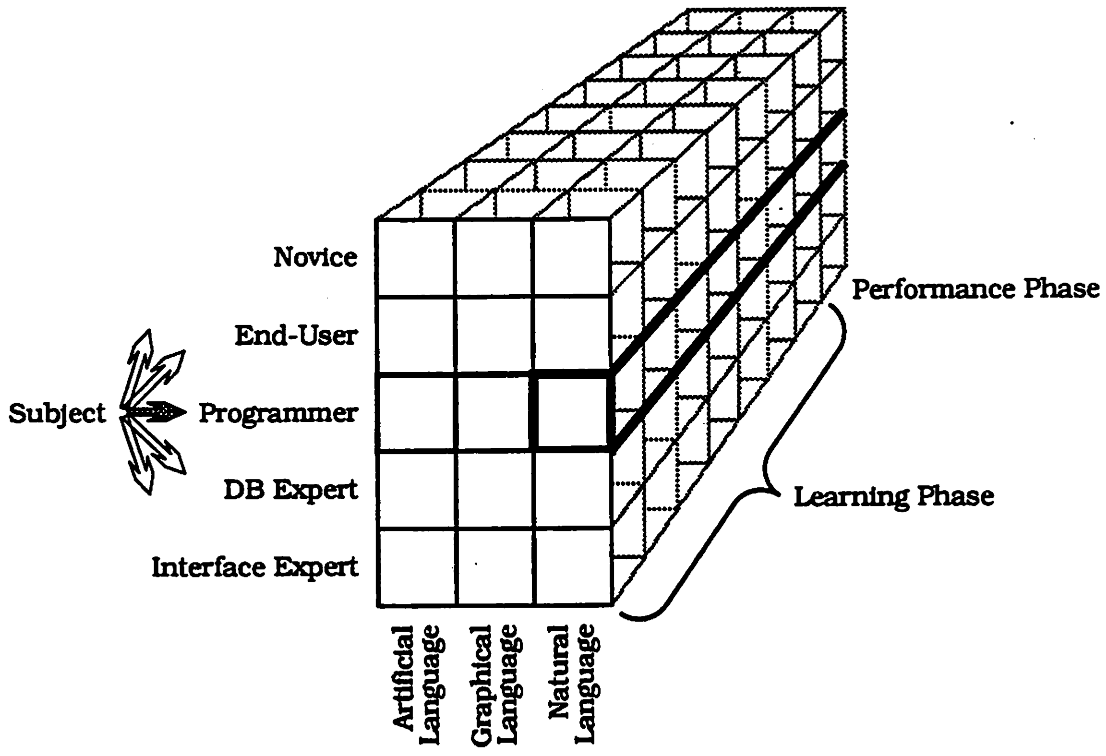


Figure 6: Experimental Design with programmers assigned to NL highlighted.

LEVEL	DESCRIPTION
1	One table, no restrictions
2	One table, one restriction
3	One table, two restrictions
4	One table, one or two restrictions, sorting
5	One table, no restrictions, aggregation (e.g., counting)
6	Two table join
7	Three table join

Figure 7: Learning phase task level definitions.

Most subjects were drawn from students at the University of California at Berkeley. They were all volunteers who responded to announcements or advertisements soliciting subjects. Each subject spent about two hours working with one of the interfaces. The subjects were classified according to prior computer experience as show in figure 8.

GROUP	DEFINITION
Novice	less than 10 hours of computer experience ever
End User	experience with applications (e.g., spread sheet or word processing), but no programming experience
Programmer	programming experience, but no database experience
Database Expert	knowledgeable in SQL or QUEL
Interface Expert	knowledgeable in the interface being used (SQL, Simplify or DataTalker)

Figure 8: Subject group definitions.

Over 80 people participated in the experiment of which 61 were used for analysis. Some subjects were used for a pilot study and others were rejected because they had experience with the particular interface. The subject distribution is shown in Figure 9. The cell "database experts and AL" is empty because a database expert was assumed to know SQL or QUEL, and only interface experts were allowed to have any experience with the interface they were learning. Hence, database experts using AL are actually interface experts because of their SQL knowledge.

SUBJECT GROUP	TOTAL	AL	GL	NL
Novice	15	5	5	5
End User	15	5	5	5
Programmer	15	5	5	5
Database Expert	10	0	5	5
Interface Expert	6	2	2	2

Figure 9: Number of subjects in each category.

The interface expert category is smaller because the variation between subjects was expected to be small and because its primary purpose was to serve as a benchmark by which to compare the performance of the other groups.

Subjects were randomly assigned to one of the three treatments: AL, GL or NL. Everything was identical about the three treatments except for the interface used and the content of the help materials provided during the learning phase which is described below.

Each subject was given a brief introduction to the experiment, the high school database, and the pictorial task presentation method. They were told they would be getting information out of the computer using a query language. After being shown a sample task, the subjects were shown how that task could be performed with the interface to which they were assigned.

Subjects then worked through the learning phase based on the procedure shown in Figure 10. Subjects were given a task beginning at level 1 which they were to perform. If they could not solve the task, help was provided in the form of written advice. Help was broken down into the four levels shown in figure 11.

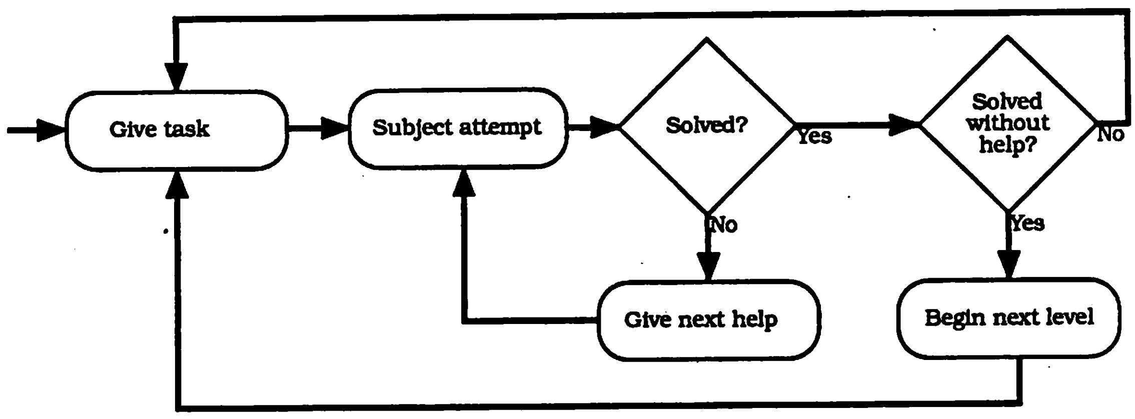


Figure 10: Treatment in Learning Phase

LEVEL	DESCRIPTION
Hint	A two sentence description of the essence of the solution.
Example	A query used to solve a similar task.
Explained Example	The steps followed to generate the example query.
Answer	A query that would solve the current task.

Figure 11: Help level descriptions.

Subjects who required help advanced to the next level when they successfully completed a task without help. Subjects who got the first task at a level correct without help were required to solve another task at that level before being advanced.

After completing all levels, subjects moved to the performance phase of the experiment. Subjects were give a series of tasks that represented query types in the learning phase as well as query types not seen before. The performance phase tasks and the corresponding learning phase levels are shown in figure 12.

PERFORM. LEVEL	DESCRIPTION	LEARNING LEVEL
1	One table, no restrictions	1
2	One table, two restrictions	2 & 3
3	One table, aggregate counting	5
4	Three table join	7
5	Two table join, one restriction, sorting	4 & 6
6	Existence	none
7	Self-referential join	none

Figure 12: Performance phase task level definitions.

The existence and self-referential join tasks were included to see if subjects were able to extend their knowledge of the interfaces to situations that had not been explicitly taught, and to see how experts on each of the interfaces would deal with complex tasks.

An existence query asks if particular instances exist. For example, find all activities that have no student participants. A self-referential join query has a join of a table to itself.

Each subject was videotaped during the experiment. Records were also kept by the experimenter (J. Bell) during each session. These records included the time at each step through the procedure (e.g., when the task was given, when help was given, when the subject moved to next level, etc.) as well as verbal comments made by the subject and difficulties the subject encountered (e.g., forgetting to clear out an old query before creating a new query). In addition, for the AL and NL treatments, a transcript file was recorded.

4. User Performance Results

The results of the performance phase of the experiment are presented in this section. Figure 13 shows the mean percentage of correct queries completed by each group on the performance phase tasks on which the subjects were trained (i.e., tasks 1-5). A value of 1.0 indicates that all subjects were able to complete that task, while a value of 0.5 indicates that only half of the subjects were able to complete it.

Novices were most successful on single table queries (i.e., tasks 1-3) with AL. The SQL queries required a very simple structure with English keywords. Novices were able to remember the commands and to substitute the appropriate table and column names. For task 1, novices did slightly better with GL, but then they did much worse on tasks 2 and 3 primarily because of the complexity of the keyboard-mouse interface. Novices were able to perform task 5 (two table join) with NL. However, task 2 (one table with two restrictions) was difficult for all user groups when using NL.

End users were most successful using GL, except for task 5 (two table join) which was again the one task where NL was better. The

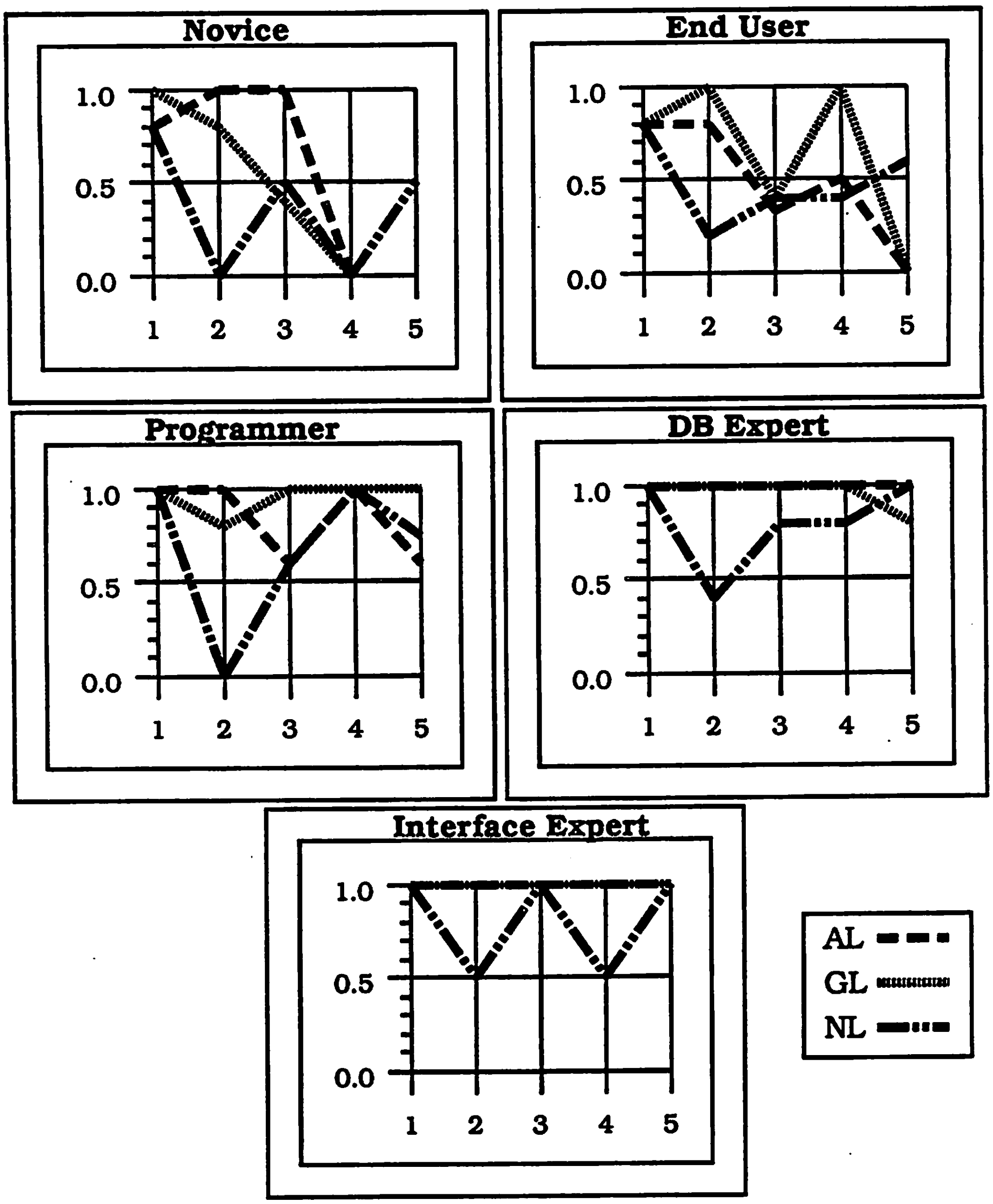


Figure 13: Mean percentage of tasks completed for trained queries.

similarity of the AL and GL graphs (e.g., tasks 2 and 4 were easier, while tasks 3 and 5 were harder) was caused, we believe, by the similarity in AL and GL operations (e.g., joins and counting). NL is below AL on task 2, above AL on task 5, but otherwise essentially equivalent.

Programmers were virtually perfect using GL. AL was worse on join queries (tasks 3 and 5). Again, NL was nearly identical to AL on all tasks except task 2.

DB experts were able to solve almost all tasks using both AL and GL. Note that DB experts for AL were the same subjects as interfaces experts for AL because of their knowledge of SQL. NL was inferior to AL and GL.

Interface experts solved all tasks with AL and GL. Even experienced NL interface users had problems with tasks 2 and 4.

Figure 14 shows the results of the untrained queries (i.e., tasks 6 and 7). NL worked extremely well for the existential task. Even novices outperformed interface experts on either AL or GL. Notice that no one was able to solve this task using GL.

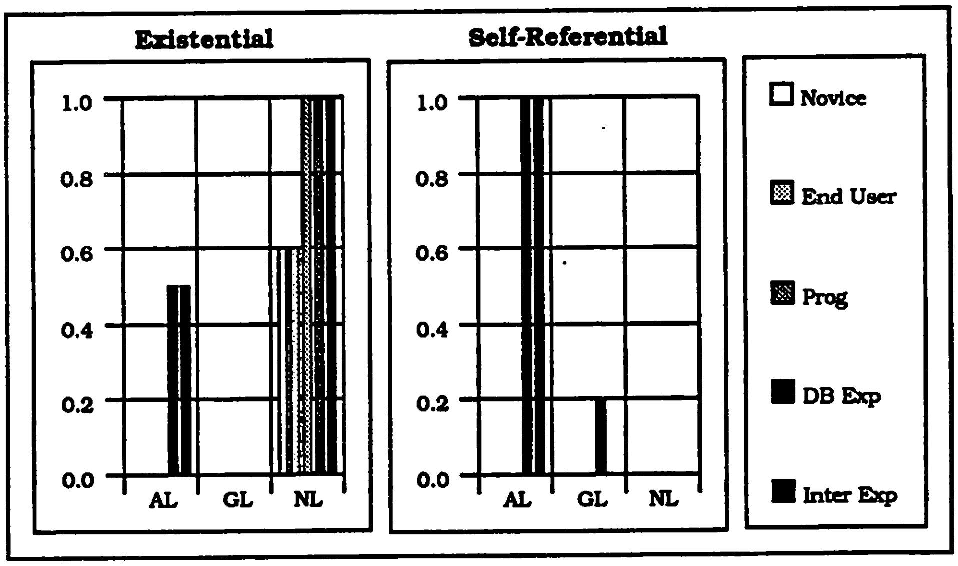


Figure 14: Mean number of tasks completed for untrained queries.

On the self-referential task, database and interface experts working on AL did the best. A few database experts were able to solve this task using GL.

5. Conclusions

We draw the following conclusions from this experiment.

1. **No interface was uniformly best across all groups or even within one group.**

Each system outperformed the others in some areas. Consequently, none of them can yet claim superiority.

2. **The best performance was achieved by experienced SQL interface users.**

Those subjects who knew SQL were most able to complete the tasks. The users' performance was most predictable and they were able to perform all of the required tasks. If users are able to dedicate the resources required to become an AL expert, then AL is likely to be the query language interface of choice.

3. **GL was better than AL on queries that GL could handle, except for novices.**

Apart from novices, the GL interface outperformed AL on single table restrictions, joins, counts and sorting. The problem with GL was that users could not figure out how to do the self-referential join task and the existential query is impossible. The novice problem was a result of the keyboard/mouse interface complexity. The three button mouse, pop-up menus, button menus, windows and dialog boxes were too much for novices to figure out while also learning the query language itself. In addition, the interface metaphor for counting confused some users.

4. NL results were mixed

NL was the only interface that subjects other than interface experts could get the existential query. Furthermore, novices were able to outperform the experts on that query. NL was also the only interface on which novices and end-users could solve any tasks that involved joins. Apart from task 2, NL was generally the same as the lower of AL and GL.

We conclude that both GL and NL show promise as a better interface than SQL. However, GL cannot handle some queries and user performance with NL is too unpredictable. Note that the results presented are for getting queries correct, and for the performance phase only. We are still analyzing the results on the time to solve a task and on the learning phase.

References

[Bell90] Bell, J., (1990). *Human Factors Evaluation of a Textual, Graphical, and Natural Language Query Interfaces.* Ph.D. dissertation, University of California at Berkeley (in progress, January 1990).

[BoBE83] Boyle, J.M., K.F. Bury and R.J. Evey (1983). Two studies evaluating learning and use of QBE and SQL. In *Proceedings of the Human Factors Society 27th Annual*

Meeting. (pp. 663-667). Santa Monica, CA: Human Factors Society.

[BrSh78] Brosey, M. and B. Shneiderman (1978). Two experimental comparisons of relational and hierarchical database models. *International Journal of Man-machine Studies*, 10, 625-637.

[Catt80] Cattell, R.R.G. (1980). An entity-based user interface. *Proceedings 1980 ACM SIGMOD Conference on Management of Data.*

[Dame81] Damerau, F.J. (1981). Operating statistics for the transformational question answering system. *American Journal of Computational Linguistics*, 7, 30-42.

[FiSB85] Fink, P.K., A.H. Sigmon and A.W. Biermann (1985). Computer control via limited natural language. *IEEE Transactions on Systems, Man, and Cybernetics*, 15, 54-68.

[GrWa78] Greenblatt, D. and J. Waxman (1978). A study of three database query languages. In B. Shneiderman, Ed.), *Databases: Improving usability and responsiveness*, New York: Academic Press.

[Jark85] Jarke, M., J.A. Turner, E.A. Stohr, Y. Vassiliou, N.W. White, and K. Michielsen (1985). A field evaluation of natural language for data retrieval. *IEEE Transactions on Software Engineering*, SE-11, 97-114.

[Krau80] Krause, J. (1980). Natural language access to information systems. An evaluation study of its acceptance by end users. *Information Systems*, 5, 297-319.

[Loch78] Lochovsky, G.H. (1978). *Data base management system user performance.* Ph.D. dissertation, University of Toronto, Canada.

[OgBr83] Ogden, W.C. and S.R. Brooks (1983). Query languages for the casual use: Exploring the middle ground between formal and natural languages. In *Proceedings of CHI '83: Human Factors in Computing Systems* (pp. 161-165). New York: Association for Computing Machinery.

[OgSo87] Ogden, W.C. and A. Sorknes (1987). What do users say to their natural language interface? In *Proceedings of Interact '87 - 2nd IFIP conference on Human-Computer Interaction* Amsterdam: Elsevier Science.

[Reis75] Reisner, P., R.F. Boyce, and D.D. Chamberlin (1975). Human factors evaluation of two data base query languages - Square and Sequel. In *Proceedings of the National Computer Conference.* (pp. 447-452). Arlington, VA: AFIPS Press.

[Shne78] Shneiderman, B. (1978). Improving the human factors aspect of data base interactions. *ACM Transactions on Database Systems*, 3(4), 417-439.

[SmWe83] Small, D.W., and L.J. Weldon (1983). An experimental comparison of natural and structured query languages. *Human Factors*, 25, 253-263.

[ThGo75] Thomas, J.C. and J.D. Gould (1975). A psychological study of query by example. In *Proceedings of the National Computer Conference*. (pp. 439-445). Arlington: AFIPS Press.

[WeSt81] Welty, C. and D.W. Stemple (1981). Human factors comparison of a procedural and a nonprocedural query language. *ACM Transactions on Database Systems*, 6, 626-649.