12

HIGHER ORDER SOFTWARE, INC.
806 MASSACHUSETTS AVENUE
POST OFFICE BOX 531
CAMBRIDGE, MASSACHUSETTS 02139

FINAL REPORT
CONTRACT NO. N00014-82-C-0137

JULY 1982

DATA BASE QUERY WITHIN HOS SYSTEMS METHODOLOGY

PREPARED FOR
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
800 NORTH QUINCY STREET
ARLINGTON, VIRGINIA 22217

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER TR-35	2. GOVT ACCESSION NO. AD-A118369	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DATA BASE QUERY WITHIN HOS SYSTEMS METHODOLOGY		5. TYPE OF REPORT & PERIOD COVERED Final Report 1 Jan 1982 - 30 Jun 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard Smaby		8. CONTRACT OR GRANT NUMBER(s) N00014-82-C-0137
9. PERFORMING ORGANIZATION NAME AND ADDRESS Higher Order Software, Inc. 806 Massachusetts Avenue Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N 42; RR042-09; RR0420901; NR 196-172
11. CONTROLLING OFFICE NAME AND ADDRESS DCASMA-Boston 495 Summer Street Boston, MA 02210		12. REPORT DATE July 1982
		13. NUMBER OF PAGES 55
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research 800 North Quincy Street Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

For public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

data-base query	systems methodology
personalized	functional data model
query language	graphics

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

Applying the theory language and tools of system specification methodology of HOS to data base query promises to yield a data-base query tool which is accessible to users with broad analytic capabilities independent of knowledge of computing.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

I. SUMMARY OF RESEARCH EFFORT

HOS consists of a methodology and a language which permits a user to specify a system which is free of a large class of errors. There are also computer tools, i.e. software, that automatically determine whether errors are present and also automatically generate code, e.g., FORTRAN to implement or simulate the system. There are many aspects of HOS which would be beneficial to specifying data-base access. It is a hierarchical, structured methodology which ensures step-wise refinement of a specification. It allows data-flow to be readily visualized, which equally well facilitates the understanding of information flow. It is general purpose, allowing completely expandable data-base applications, starting with and going beyond query capabilities within a uniform language and methodology. The underlying kinds of objects in the methodology, called data types, are characterized by the operations that apply to them, which correspond to the primitive data-base access operations. The methodology requires that specifications be broken down into layers of naturally co-occurring operations which are appropriate to the topic under discussion, and it sharply distinguishes specification of a system from implementations of the system, and thereby can provide data models of data-bases that are completely independent of implementation considerations.

To provide means of utilizing the HOS methodology and language to support data-base query (and eventually other data-base manipulation) was a worthwhile challenge. We defined that challenge in the requirement of providing a data-base query capability which was

1. Based on the HOS methodology

2. Personalized to allow the user to specify his or her conceptual view of the data-base as well as specify arbitrary utilities, to the extent he or she was knowledgeable about the utilities

3. Graphical to allow a helpful visualization of information flow as well as of the hierarchial structure of the concepts being defined

4. Interactive, both in providing for quick entry of specifications of queries and utilities, and in providing feedback at the user's request on possible errors in the specification, at any point during the construction of the specification, and necessarily prior to any code generated to support querying

5. Complete, in that the language could express an arbitrary query.

These requirements are met in part by prior existing HOS tools. However, significant effort went into

(1) Designing languages which might better fit the task of data-base access than the current HOS syntax, but would be compatible with it, showing they are complete, and implementing one of them

(2) Designing an interface to make what have been system software development tools into a tool usable by anyone who has a good understanding of his or her knowledge domain (which is completely independent of any knowledge of the internals of the data-base or computer programming)

(3) Developing a concept of data-base data model appropriate to the HOS methodology, as well as support utilities for defining and populating such data models.

In order to evaluate the potential of combining a general systems methodology with data-base query, we designed, implemented and ran an experiment, using the utilities developed. The efforts in direct support of the experiment consisted in

(1) Choice of variable that would be most helpful in evaluating the potential of using a structured methodology in data-base query

(2) Designing the experiment protocol around that variable, in particular the participant's choice of an option to build supporting operations for his or her querying

(3) Construction of a set of queries written in English that would produce results which were differential with respect to the variable

(4) Defining a specific data model to simulate a personnel data-base and populating that data model with enough data to support the set of queries

(5) Running the experiment in a realistic environment, in which the participants, with general record-keeping and other office skills were obtained from a temporary service agency, and used graphic terminals to enter their construction of the queries

(6) Analysis of the results of the experiment.

The general conclusions resulting from the research and development effort described above are

(1) There is indeed a nice fit between the HOS system methodology and data-base design and manipulation; in fact, the HOS methodology encourages data models which are more accessible, since it requires complete independence of implementation considerations

(2) There are complete query languages derived from the HOS language and formally reducible to it

(3) A sharp division between the performance of two groups of subjects suggests that the particular system tested is more suitable for someone with good general analytic experience rather than someone who is experienced in only performing routine activities.

A. LANGUAGE DESIGN

1. Motivation

a. Background on HOS

The HOS language, called AXES, was designed as a language in which to express correct systems specifications, which were completely precise. In fact, specifications in the language, AXES, allow the automatic generation of code, if the system being specified is a software system.

AXES is a hierarchically structured language, expressing directly the stepwise refinement or definiton of a function. For example, one could specify a simple function called

SEQADDMUL

by defining it in AXES as in Figure 1.

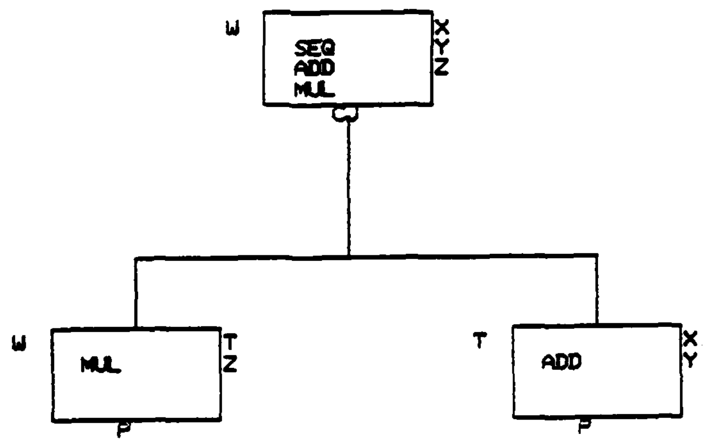


FIGURE 1

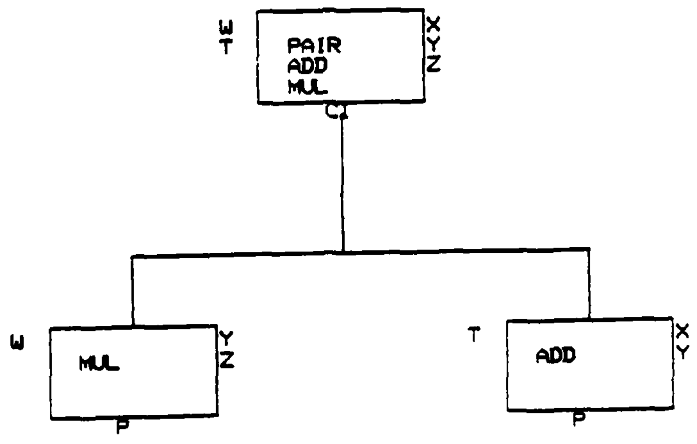
Figure 1 is understood as saying that the ADDMUL function is composed out of the ADD and MUL functions, performing the ADD function first and using its output as input to the MUL function along with z.

The points to note are:

1) data is referred to with variables.
2) data flows from right to left, i.e., inputs are on the right and outputs on the left, derived from the mathematical notation: $y = f(x)$.
3) functions are represented by boxes.
4) the manner of combining the offspring is redundantly specified using a label, e.g., CJ for co-join, at the bottom of the parent box.
5) some functions are labelled as primitives with P or operations with OP to indicate respectively that no further decomposition is to be performed or that further decomposition has already been performed elsewhere and stored in a library of available operations.

Item 4 needs a bit of clarification. According to the HOS methodology, a function is analyzed (or decomposed) in one of three ways. The first is as in Figure 1 where the right offspring feeds some data to the left, thereby "joining" the two offspring via shared data, i.e., data produced by the right and consumed by the left.

The second type of decomposition is used when the offspring are merely included as two parts of the parent process without one feeding data to the other, as shown in the example of Figure 2. The details of the methodology are available in [1,2].



Such a decomposition is called a co-include and abbreviated CI.

The last type of decomposition is choice between one function or another based on a boolean valued variable. Figure 3 shows an example.

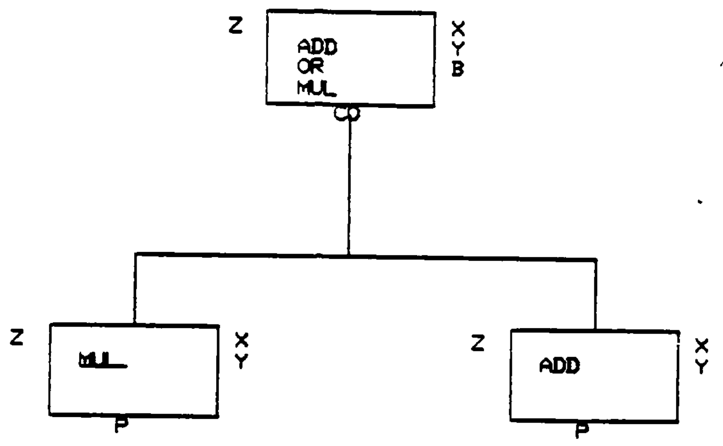


FIGURE 3

The decomposition in Figure 3 is called co-or, abbreviated CO.

There are strict rules of data flow for each of the three decomposition structures.

While the extra effort required by the methodology is excessive when specifying a simple arithmetic computation, it turns out to be richly repaid when specifying large programming projects [3]. It is also useful for designing smaller programs to support queries, which we will have more to say about later.

b. Relational and Functional Query Languages

The various relational and functional query languages that have been designed, e.g., SEQUEL, QBE, and DAPLEX have important features to profit from in the design of any new query language. SEQUEL and FQL have a very logical structure allowing for unambiguous construction of complex queries. The English question:

"What is the age of Marilyn Brewer?"

might be formulated in SEQUL as

```
SELECT  AGE
FROM    PERSONNEL
WHERE   NAME = "MARILYN BREWER"
```

which is based on an underlying data model with entity type PERSONNEL, say

PERSONNEL

NAME	UNIT	JOBID	SEX	CITY	AGE	YEARS OF EXPERIENCE	PAYRATE	NO. OF CHILDREN	SUPERVISOR

In DAPLEX, the same query could be expressed as

```
FOR EACH PERSON
     SUCH THAT
          NAME(PERSON) = "MARILYN BREWER"
     PRINT  AGE(PERSON)
```

More complex queries are logical extensions of simpler queries. For example,

"I need a list of the persons and their sex for people who have jobs for which the base payrate is less than 12000."

For the relational data model we would need to have, in addition to the personnel entity type above, a job entity type, such as

JOB

JOBID	BASE PAYRATE	MINIMUM EXPERIENCE

In SEQUEL we could query as

```
SELECT NAME, SEX
FROM PERSONNEL, JOB
WHERE PERSONNEL.JOBID = JOB.JOBID AND BASE_PAYRATE<12000
```

While in DAPLEX it might be

```
FOR EACH PERSON
     SUCH THAT
          BASE_PAYRATE(JOB(PERSON))<12000
     PRINT NAME(PERSON)
```

PRINT SEX(PERSON)

The query language: QUERY_BY_EXAMPLE or QBE for short has a nice feature of allowing forms type entry of a query. The first simple query above could be entered in QBE as

PERSONNEL

NAME	UNIT	JOBID	SEX	CITY	AGE	YEARS OF EXPERIENCE	PAYRATE	NO. OF CHILDREN	SUPERVISOR
M.B.					.P				

While QBE provides a convenient interface for simple queries, more complex queries become very unclear.

We attempted to combine the good features of each of the above languages in a single language and query system, which in addition conformed to the larger methodology of HOS.

2. Designs

We considered a number of language designs all deriving ultimately from two different syntaxes for AXES, the first of which was briefly described in the preceeding section, and the second of which has a more graphical presentation of data flow, as in Figure 4, corresponding to Figure 1 in the previous section. In this latter syntax, the arrows serve to identify the data rather than using variables as in the previous section.

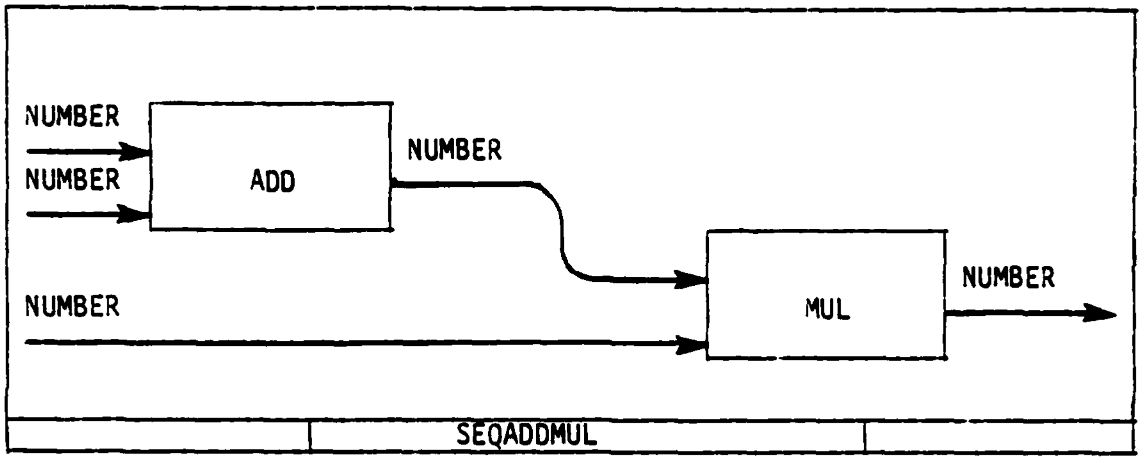


FIGURE 4

While either of the basic syntaxes are capable of specifying any computation or query, they lacked certain important conveniences for use in a querying task, most importantly quantifiers such as <u>ALL</u> or <u>SOME</u>, and perhaps adjectives.

The two designs settled on are logically identical, but graphically different. Both are based on a functional data model, quite similar to that used by DAPLEX, i.e., data is manipulated using functions basic to the particular data-base, rather than thinking in terms of record types or entity tuples.

We can exemplify the two designs by looking at how they express the sample queries above. The query

"What is the age of Marilyn Brewer"

could be expressed in the first as in Figure 5.

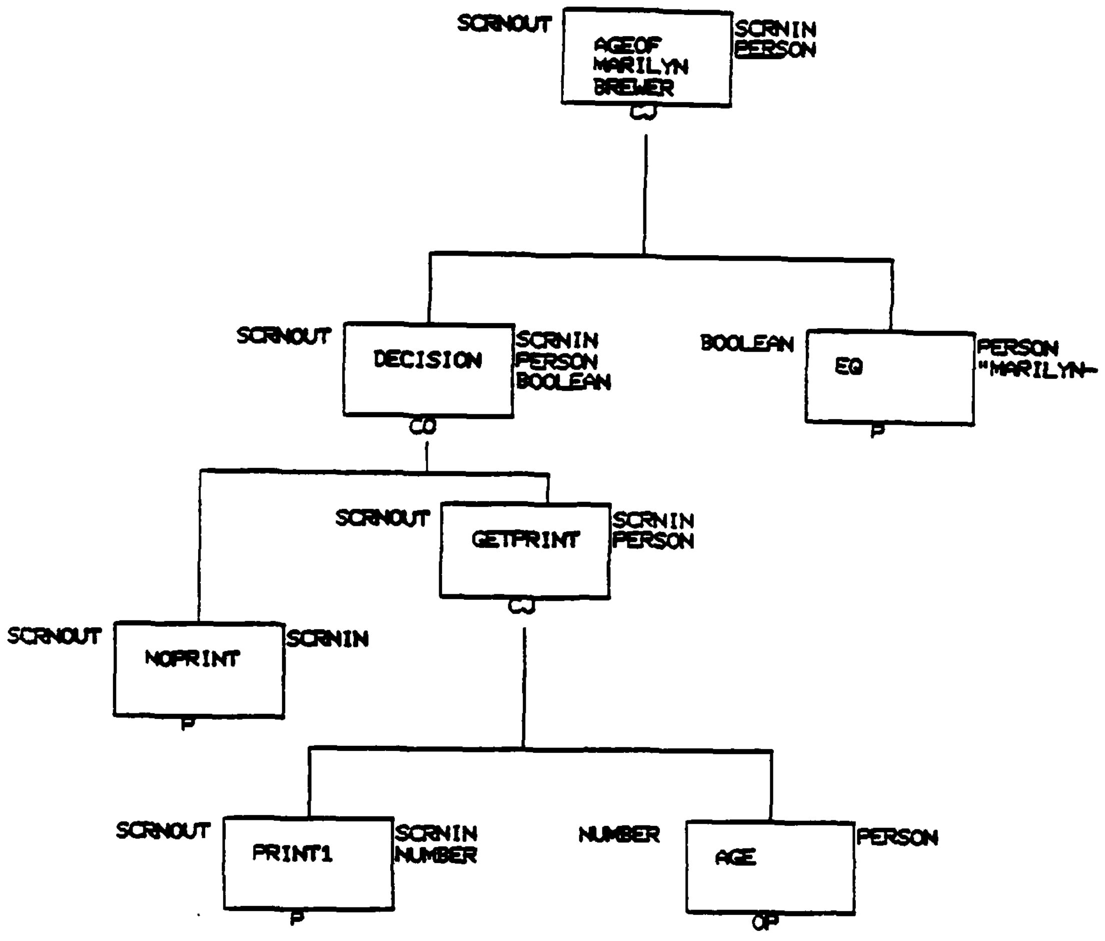


FIGURE 5

The sense of Figure 5 is that if a person (or person's name) is equal to MARILYN BREWER, then the AGE of that person will be obtained and printed as an update to the screen.

Most queries will tend to have the same format, shown in Figure 6.

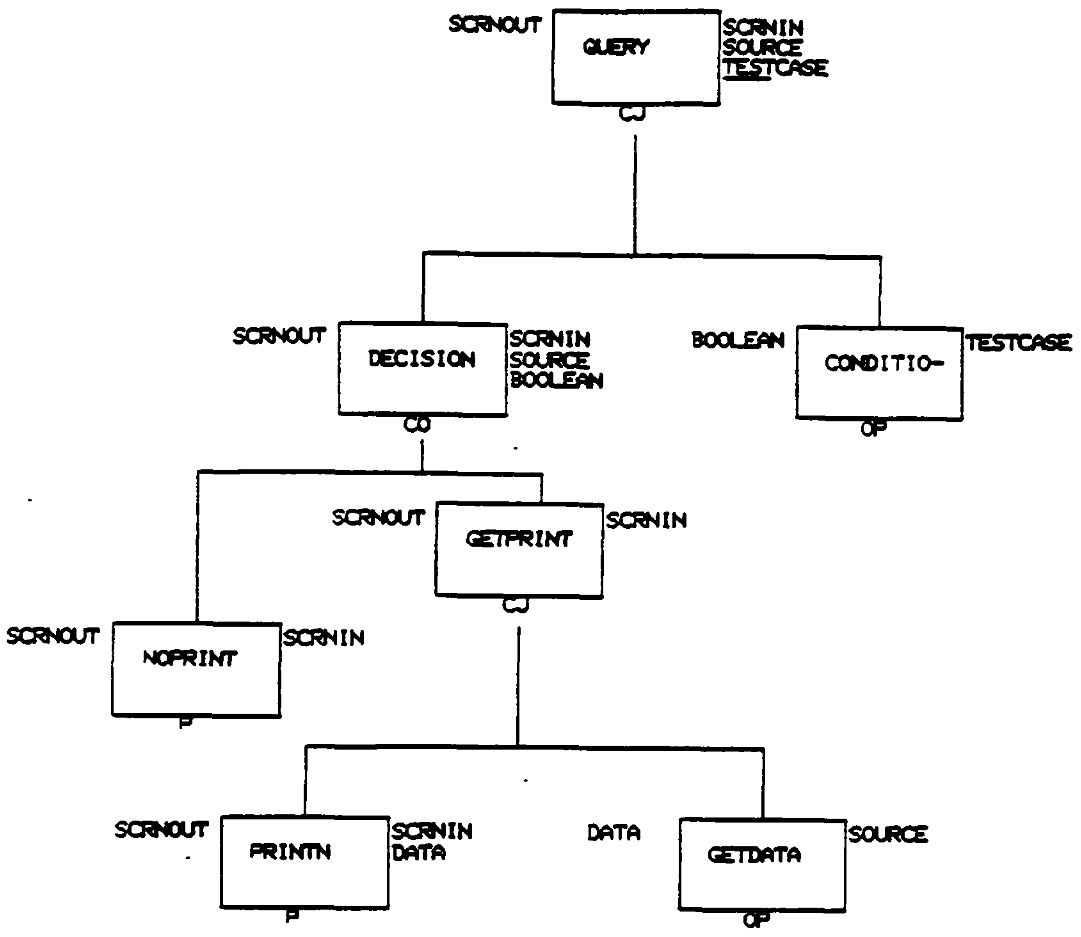
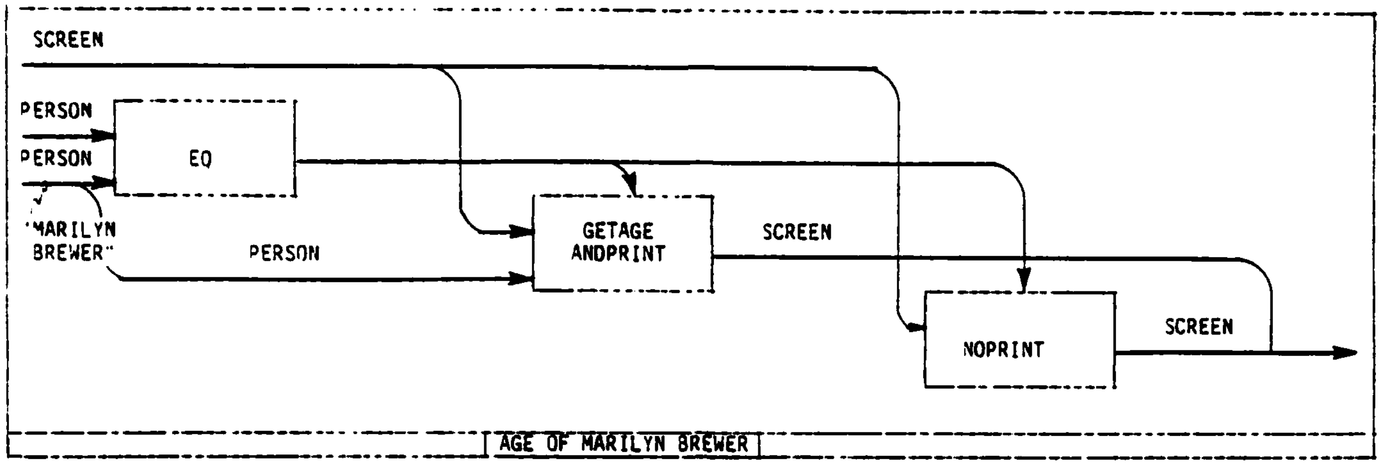


FIGURE 6

The CONDITION and GETDATA will vary in complexity but the remaining framework will stay the same.

Since the whole point of a query is to provide a representation of data on a screen for a user the alteration of the screen is indicated by the transformation of state of the screen before the case (SCRNIN) to the state after the case (SCRNOUT).



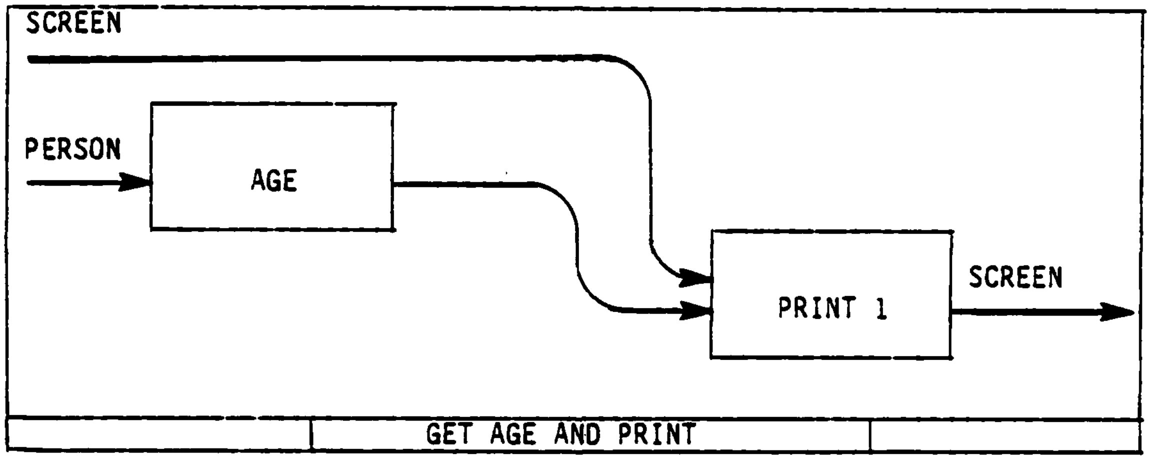


FIGURE 7

The expression of that same query in the second language is shown in Figure 7.

The second, more complex example appears in Figure 7.

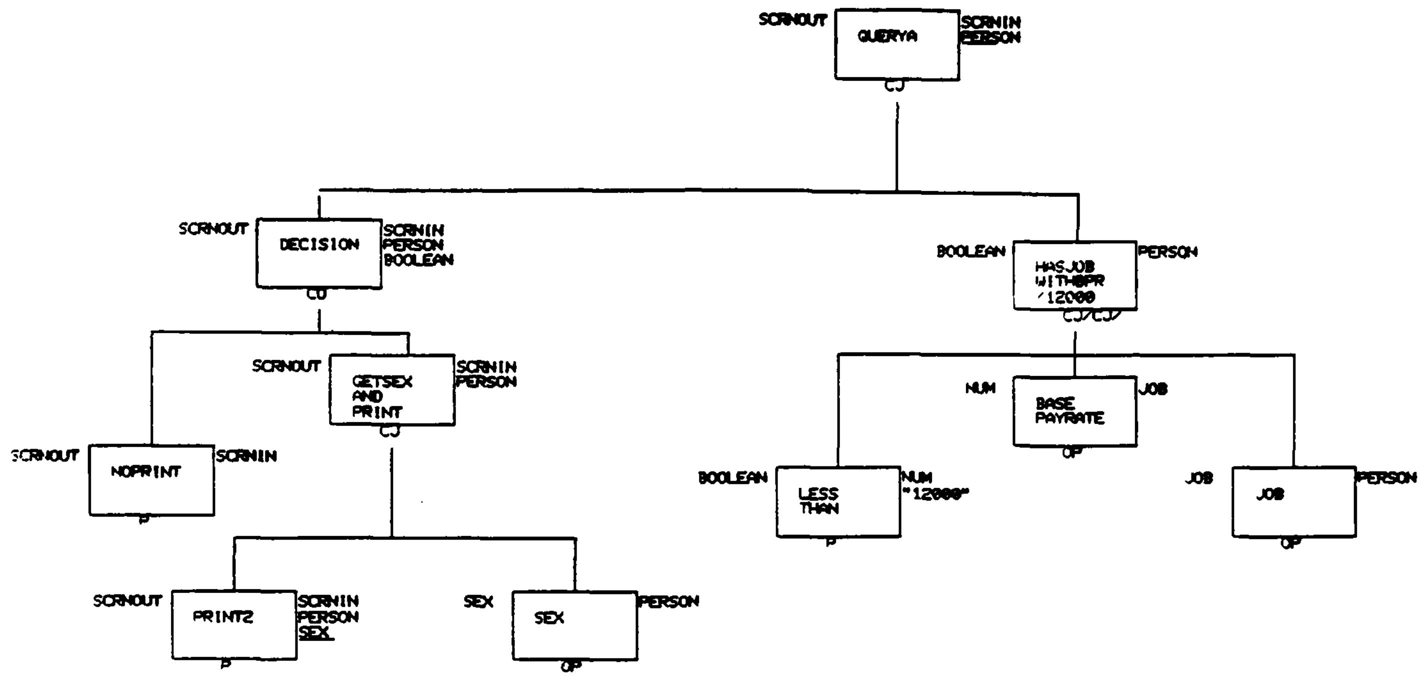
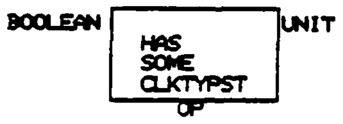
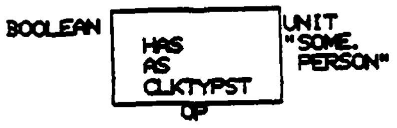


FIGURE 8

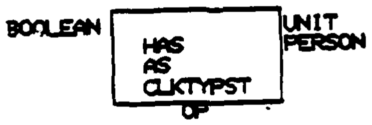
The introduction of quantifiers into the syntax of the languages is quite straight-forward. Quantifiers can only be used on conditions, i.e., functions which produce a boolean value: true or false. For example, suppose we wish to identify units which have some clerk typists; i.e., we want a function



which given a unit produces true if the unit has at least one clerk typist and false otherwise. This function can be expressed using the quantifier SOME as



which is based on the operation



which is in turn defined as in Figure

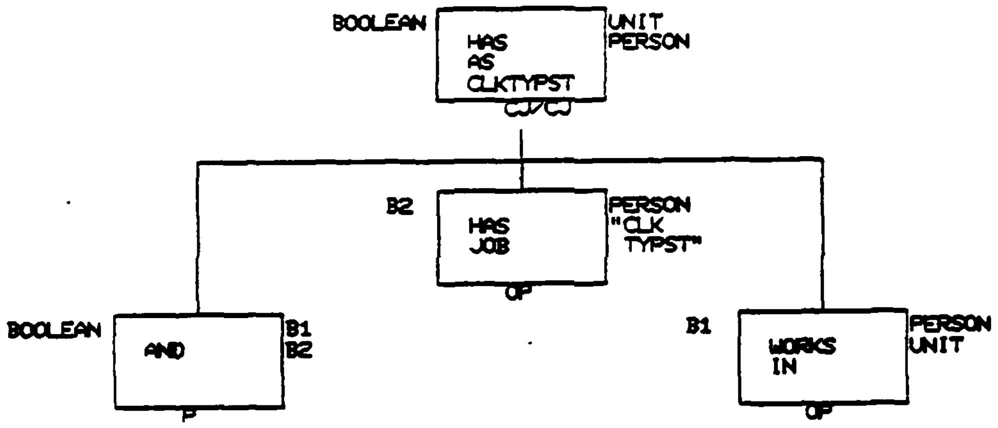
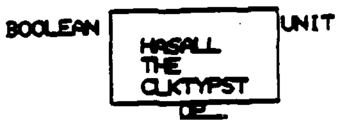
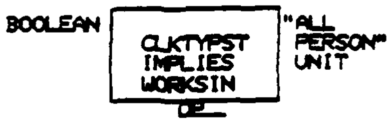


FIGURE 9

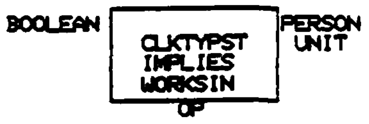
The concept of a unit having all the clerk typists is similarly analyzed



is specified



which is based on



which is defined as in Figure

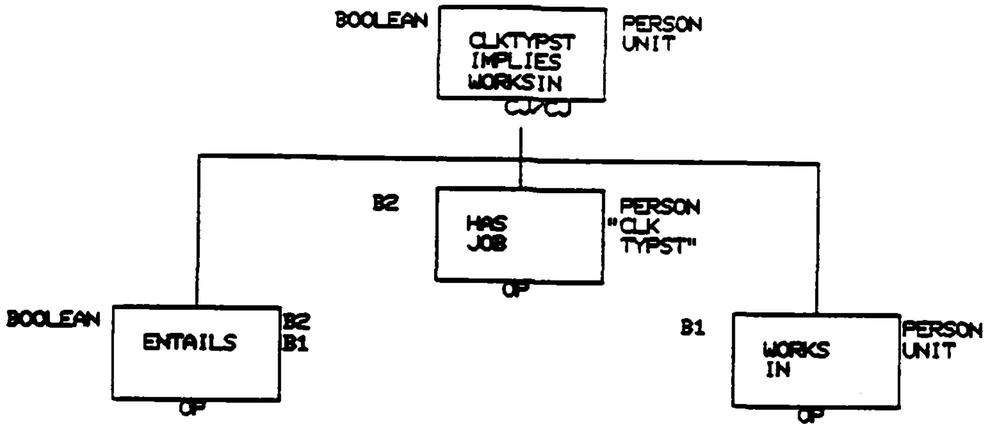


FIGURE 10

3. Interfaces for a Variety of Users

We considered various interfaces for the data-base query system. We wanted to provide for as many kinds of users as possible under the same language. There are two broadly defined classes of users of data-base information: those users who perform routine data-entry and retrieval and those who devise new views of and new operations on the information in the data-base.

The strength of the HOS methodology lies in supporting the latter kind of person. However, the persons performing routine tasks can use the tools devised for them by the latter people.

The more analytic user can construct generalized queries, e.g., a query which is a generalization of the one in Figure

7. Rather than having to construct a whole new specification of a query when the question is:

> "I need a list of the persons and their sex for people who have jobs for which the base payrate is less than 22000",

the analytic user can create a general query which varies over the rate. The analytic user can also extend the data-base operations by defining some of his or her own operations which can be used repeatedly to make querying more economical. The next two figures, 11 and 12, show these points.

The analytic user creates a generalized query which can be used by a routine user.

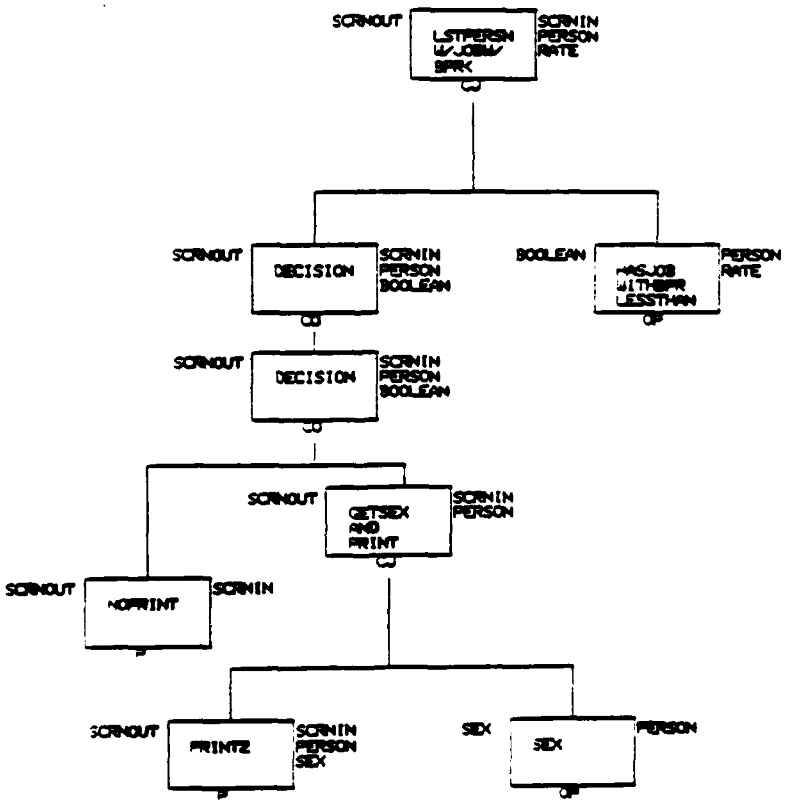


Figure 11

The analytic user also creates a defined operation to support the above query as well as other queries.

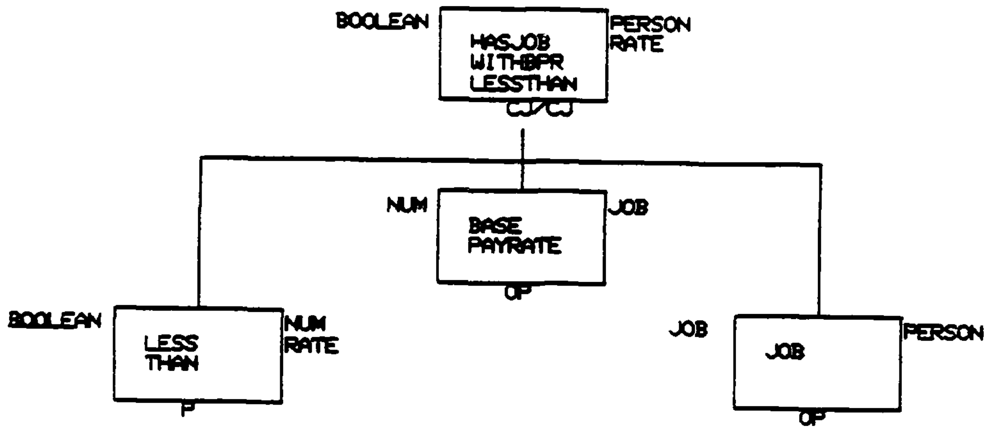
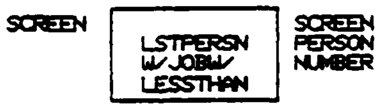
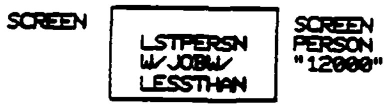


FIGURE 12

The generalized query can be provided to the routine user as a top node only



The routine user can replace the number slot by an actual number, e.g.,



and immediately have the query run on the data-base.

B. THEORETICAL LANGUAGE EVALUATION

The languages designed were evaluated according to certain theoretical properties (as well as human factors).

1. Completeness

The query languages were easily shown to be complete by virtue of the existence of quantifiers and the various boolean functions.

The accepted test of completeness for expressibility or definability of queries in a query language is whether it can express any arbitrary predicate or relation expressible in first order quantificational calculus.

Consider an arbitrary query formulated is a relation in first order logic:

$$\varphi(X_1, \ldots, X_n)$$

The set of all n-tuples satisfying that relation will be printed by using a scheme similar to the one described earlier, shown in Figure 13.

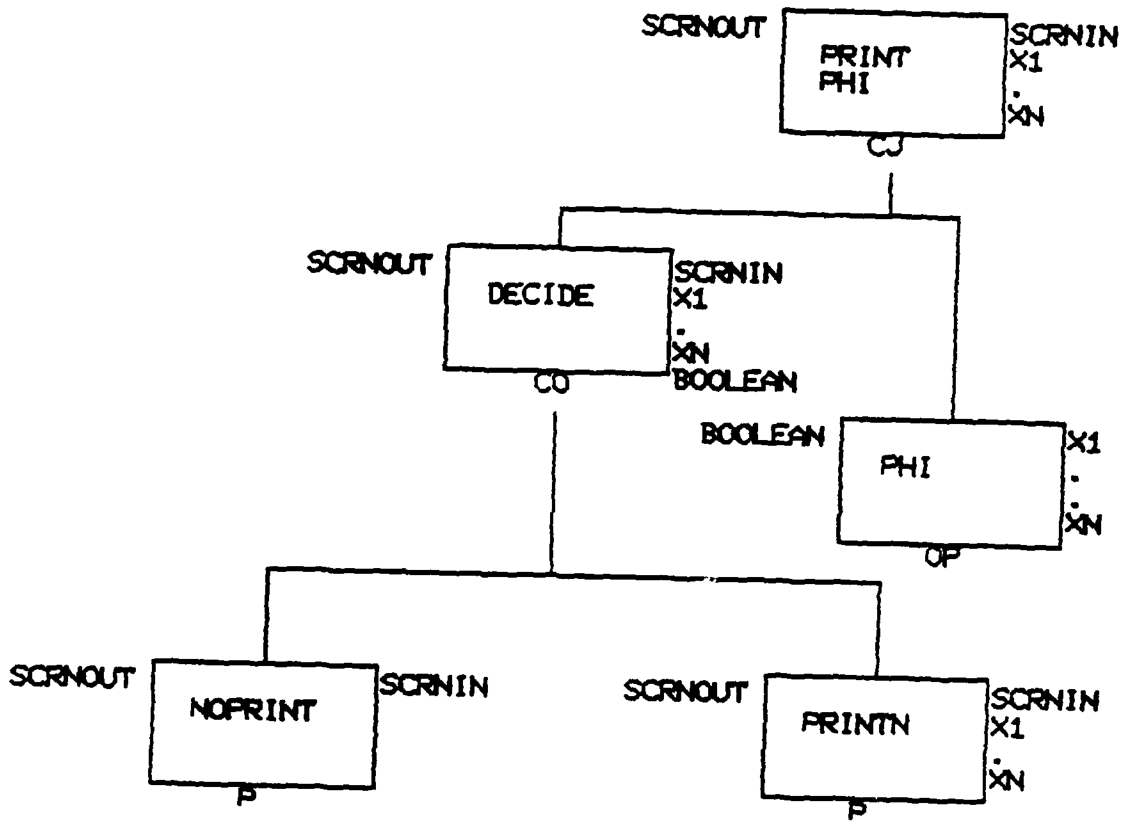
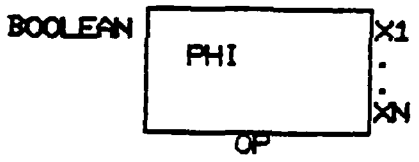

SCRNOUT
PRINT PHI
SCRNIN
X1
.
XN
CJ
SCRNOUT
DECIDE
SCRNIN
X1
.
XN
BOOLEAN
CO
BOOLEAN
PHI
X1
.
XN
OP
SCRNOUT
NOPRINT
SCRNIN
P
SCRNOUT
PRINTN
SCRNIN
X1
.
XN
P

FIGURE 13

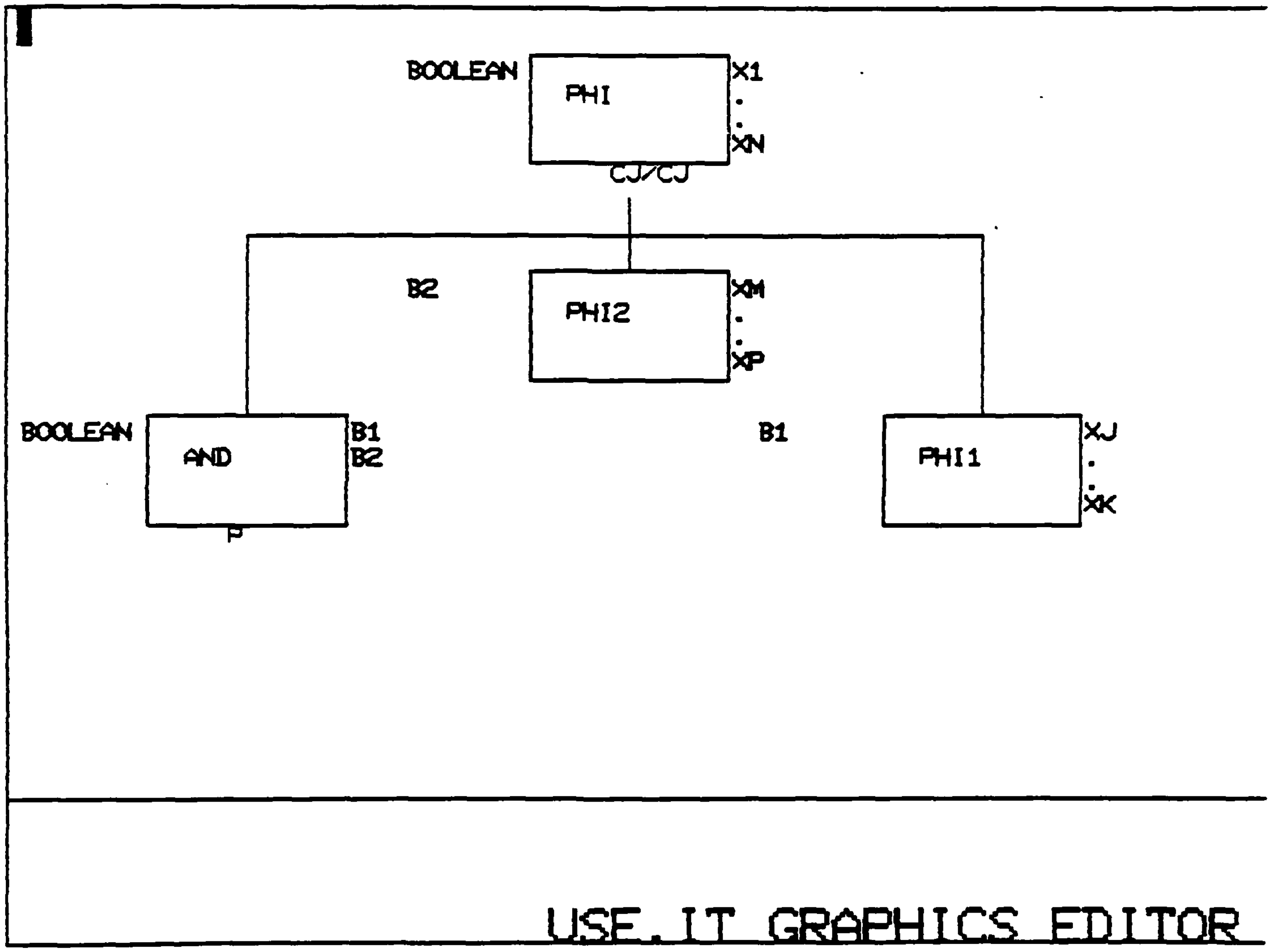
Now $\varphi(X_1,\ldots,X_n)$ may be a complex expression of first-order logic; that is, it could be a boolean combination of other relations or it could be a quantification of another relation. Rather than give a complete and formal proof, we present a couple of sample cases in the proof which should make how to carry out the complete proof rather obvious.

Suppose $\varphi(X_1,\ldots,X_n)$ is a boolean AND of two sub-relations, $\varphi_1(X_i \ldots X_k)$ and $\varphi_2(X_m \ldots X_p)$, i.e.

$\varphi(X_1\ldots,X_n)$ is $\varphi_1(X_i \ldots X_k)$ AND $\varphi_2(X_m \ldots X_p)$ then the operation

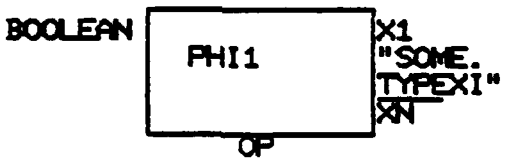


is further decomposed as


BOOLEAN
PHI
X1
.
.
XN
CJ/CJ
B2
PHI2
XM
.
.
XP
BOOLEAN
AND
B1
B2
P
B1
PHI1
XJ
.
.
XK
USE.IT GRAPHICS EDITOR

For the quantifier case, suppose $\varphi(X_1,\ldots,X_n)$ is the existential quantification of $\varphi_1(X_j\ldots X_k)$, i.e.

$$\varphi(X_1,\ldots,X_n) \text{ is } (\text{SOMEX}_i\ )\quad \varphi_1(X_j\ldots X_k)\ ,$$



2. Reduction to HOS

It is important to show how the extension with quantifiers reduces to the underlying semantics of HOS. This reduction can be shown in part by giving a scheme for translating a node with a quantifier into a more complex structure without that quantifier. Those schemes were constructed.

It remains to be shown that the HOS structure resulting from the reduction truly captures the meaning of the quantifiers. It can be shown that it does for finite domains, though not for infinite domains. Fortunately, we should only be quantifying our finite domains in data-base querying.

3. Naturalness of the Functional Data Model

At some point one must cease decomposing functions into more primitive functions. The question then arises as to how the primitive functions, which are not further decomposed, are specified. According to the HOS methodology, the primitive funtions are specified by setting up axioms relating the primitive functions to each other. A related question is how does one know when to stop decomposing? The methodology requires that the primitive functions and data types of a given system belong to a natural domain of knowledge; or put in the negative, that specification types and functions not be mixed with implementation types and functions.

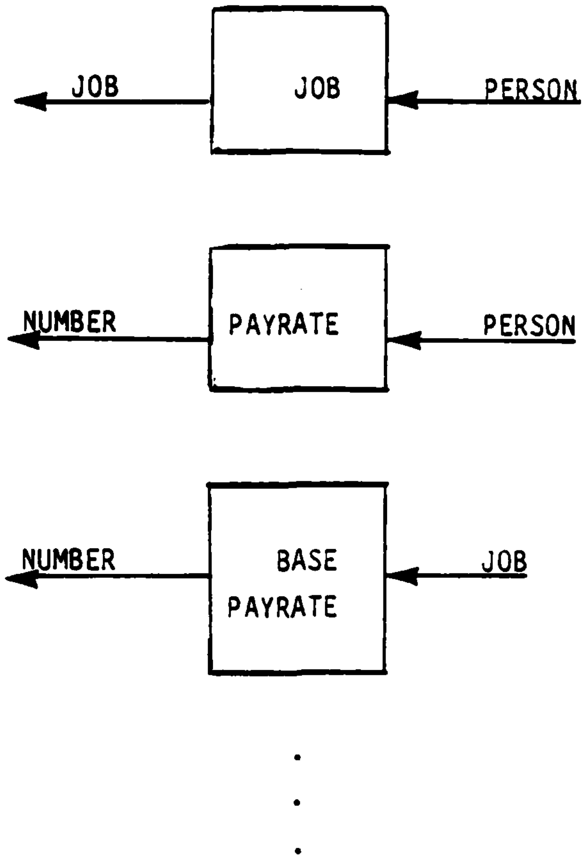
a. Axioms and Data-base Theory

The notion that the primitive functions of a system be specified by writing axioms employing them carries over nicely to the specification of a data model for a data-base. The DATA BASE FUNCTIONS figure in the appendix shows the data model that was used for the simulated data base of the experiment which was run. The figure has three aspects.

1) It shows the different types of objects used in the personnel conceptual domain as unenclosed words:

PERSON
UNIT
JOB
SEX
CITY
GROUP
NUMBER

2) It shows the set of primitive functions of the data-base enclosed in boxes, together with the types of their inputs and outputs:



(3) It shows the potential inter-connections of data flow among functions to aid in the design of queries.

Alongside such a data model diagram must be a set of axioms which specify the constraints on the primitive functions. For example, it may be invalid to have information in the data-base which shows the BASE_PAYRATE of the job held by a person being greater than the actual PAYRATE of the person. We would in that case set up an axiomatic constraint. That constraint can also be expressed in the graphical syntax being discussed as in Figure 15.

This axiom would automatically generate code which would be used to check the integrity of the data-base.

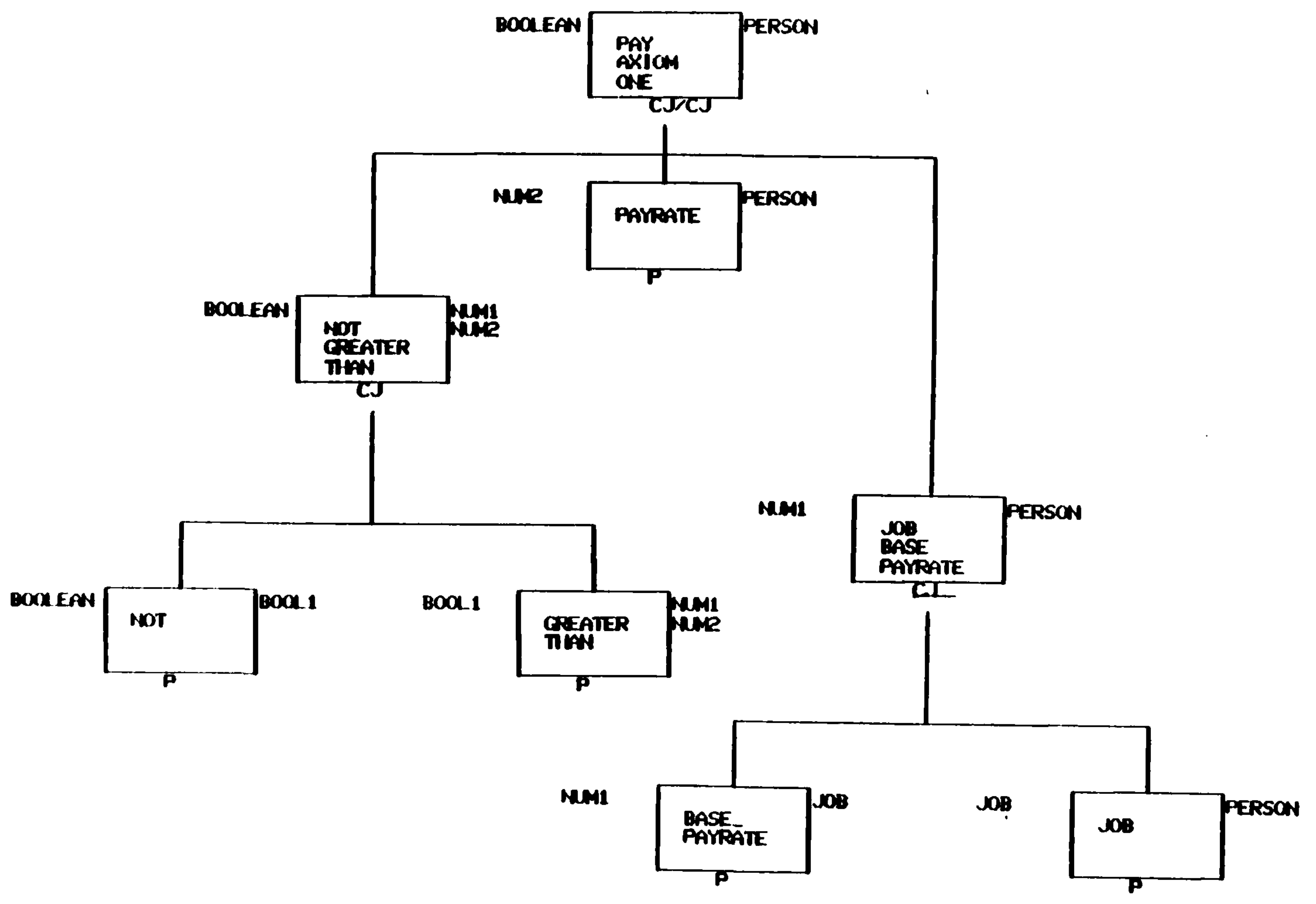

BOOLEAN
PAY AXIOM ONE
PERSON
CJ/CJ
NUM2
PAYRATE
PERSON
P
BOOLEAN
NOT GREATER THAN
NUM1
NUM2
CJ
NUM1
JOB BASE PAYRATE
PERSON
CJ
BOOLEAN
NOT
BOOL1
P
BOOL1
GREATER THAN
NUM1
NUM2
P
NUM1
BASE_ PAYRATE
JOB
P
JOB
JOB
PERSON
P

FIGURE 15

b. Layers of concepts

The specification of a system must be sharply separated from the implementation of that system. Someone who deals with a given system conceptually, e.g., a personnel system should be able to express their concepts entirely within the domain of concepts natural to the realm of personnel and never have to consider how a computer might implement those concepts. For example, the personnel manager should not have to know whether the PAYRATE function is stored in the same flat file as the YEARS_OF_EXPERIENCE function, or even that they are part of the same entity-tuple, which is merely a slightly more abstract formulation of the same implementation (or data) independent, something even the relational data model cannot claim.

Of course, there are reasons for modelling implementation considerations as well, but they should not affect the end-user's view of the data-base. For the end user, a functional data model is the most appropriate.

The HOS methodology requires that the specification of implementation of one system's concepts be done on a separate layer. For example, the PAYRATE function could be further decomposed into various select and project operations on a relation but this decompostion would be hidden from the user of the system on the higher layer, the personnel layer.

4. Generality and Automated Support

The pilot effort emphasized the integration of the query capability into the more general HOS software system development methodology. This methodology is supported by software tools:

1) A graphics editor which allows the efficient entry of the tree structured specifications

2) An analyzer which will check and comment on the logical correctness of a user's specification when the user wishes

3) A resource allocation tool which will efficiently allocate and reallocate memory resources in the generation of FORTRAN code

We wished to bring these tools to bear on the data-base query tasks. Minor modifications to the graphics editor, together with the development of host software, allowed the existing graphics editor to become the graphics interface for creating the editing query specifications as well as the medium of reference for communication of the analyzer to the user.

The graphics capability is, thus, being used to support the creation of correct specifications (and derivatively software) of queries and of utilities to support querying, and not for reporting the information obtained by the query. Reporting is simply done in a column format on the screen.

Since the language used for query tasks is a straight-forward extension of the HOS systems language, the entire capability of the system software development tools can be used to support querying or other data-base manipulation.

II. DESCRIPTION OF EXPERIMENT AND EXPERIMENTAL SUPPORT

We attempted to provide as realistic a setting for the testing of the query capability as possible. So, in addition to the ˉ nuage design, we implemented the integration of the first language into the existing tools, including the use of the graphics editor. The participants, thus, needed training in the use of the command keys of the editor.

We wished to test primarily the feasibility of having a query language as part of a general systems language. So we designed the experiment to evaluate the extent to which novices could understand and make use of a structured methodology in the data-base query setting. The participants, thus, also need some instruction in the HOS methodology. We planned to measure the correlation between the extent to which a participant made use of system features, especially the construction of utilities to support their querying and their degree of success at the experimental task.

A. Experiment Support Development

In the pursuit of providing a realistic task environment, we designed and populated a personnel data-base, and designed and implemented a user-interface to lead the participants through the cycle of specifying utilities and queries and posing the correct queries.

1. Data-base Simulation Support

We used the concept of a functional data model as a theoretical basis from which to design a partial personnel data-base. The functional model of this data-base is shown in Figure .

We designed and implemented utilities to support the creation and population of a functional data model. These utilities set up files based on information about the number of inputs and whether a function is single valued or set-valued. Then existing data-typing facilities were used to assign types to the inputs and outputs of the created functions, so that combination of functions incorrectly according to type would produce a response from the analyzer, which checks proper data type interfacing, proper structuring of specifications, and completeness of a specification.

The data-base was populated with sufficient values to allow reasonable answers to the queries in the experimental task.

2. <u>Query Environment</u>

In order to support the participants in their specification of queries and their automatic implementation, we designed and implemented a query environment which consisted in

1) a master system which set up and obtained tuples from the appropriate domains

2) an automatic bridge between the master system and the particular query

3) a command language user-interface which provided

    a) a choice of two modes:
    to form a query,
    or to build a utility operation

    b) a prompted cycle through the options

c) automatic handling of code generation and compiling, completely hiding any features of FORTRAN or system file-handling from the user

d) automatic expansion of the functional data-model to include operations built by the user,

e) automatic insertion in an object library, of the object code derived from an operation built by the user

4) a translation system that expanded any specifications with quantifiers to the basic HOS language.

B. EXPERIMENTAL DESCRIPTION

Each participant had general recordkeeping and other office skills and was recruited from a temporary agency for a single session and paid for his or her participation. The session lasted approximately eight (8) hours, with a half-hour rest for lunch and other short breaks as needed. The subject spent the experimental session seated under normal illumination in a room using a DEC VT100 terminal (modified for graphic capability by Retro-Graphics).

During the recruitment, potential participants were told that their task in the experiment was to learn and use a new data-base query language, which the company (HOS, Inc.) is evaluating.

Upon arrival, a participant was given a brief verbal introduction to the experiment and its structure and then given a short user's guide for the data-base query system (c.f. appendix). After the participant read this, the experimenter demonstrated (at the graphics terminal) the use of the query system. He gave a brief introduction to the HOS methodology. He showed the participant how to translate English questions about the data-base into queries in the data-base query language. After a few examples, the subject attempted some sample queries him or herself under the supervision of the experimenter. The experimenter showed both query and build modes. When the participant had worked through the practice queries, the experimenter left the participant alone for the rest of the experiment answering questions only about which keys produced a desired effect on the screen and the interpretation of analyzer messages.

The experiment was divided into three blocks, each with a fixed set of 12 queries for the participant to formulate. The participant was told that there were three sets of queries to be answered, but that it was not expected that he or she will finish all of them during the session. Participants were encouraged to enter into the computer as many correctly-formulated queries as possible in the allotted time.

At the start of each block, the participant was given a list of 12 queries to be formulated, and told that they could work on them in any order they wished and using any strategy of QUERYING and BUILDING they wished. Within each block of 12 queries there were two groups of 4 queries that were test items and one group of 4 items that were fillers. Each group of 4 test items contained a common concept with the other sentences of that same group. It was this common concept that could trigger the use of the BUILD option. The twelve sentences in each group were randomized.

The training session took approximately five hours in each case and the actual testing three hours.

At the end of the eight hours, the experimenter debriefed the participant and arranged for payment.

C. DATA ANALYSIS

The task turned out to be more difficult than expected, with no participant attempting more than 6 queries. The participants differed greatly in their accomplishments as well, with 3 of the 6 unable to achieve near successful formulating of even a single query.

The following tables show how accurate the queries were that each participant seriously attempted. Participants are identified by an initial. Correct means that the query was formulated correctly in every respect. Minor Syntax Errors were mis-spellings of names and other errors which otherwise showed correct conceptualization of the problem. Minor conceptualization errors were such things as writing "male" instead of "female", or specifying "less than" where "less than or equal to" is required. These 3 classes of queries were basically well-formed. Moderate errrors were those where it was clear what the participant intended, but one or more errors showing conceptual difficulty appeared. Severe errors were those which showed major flaws in the participant's reasoning. One participant wrote defined operations but did not seriously tackle their parent queries. This explains the last category. The number of queries in each entry which invoked a defined operation is shown in parentheses following it.

Tables I, II and III break down the data differently. Table I categorizes the accuracy of the main query only. Table II considers also the accuracy of called defined operation. (A query is only as good as its weakest link). Table III evaluates the defined operations separately.

As mentioned before, three participants were essentially unable to do the task. Of the other three participants, every query they wrote was in one of the top three accuracy classes indicating that they "had the right idea". (Participant H's incomplete queries do not indicate errors, only a lack of time to finish). It is interesting to note that despite encountering other difficulties with the syntax they were presented, all three of these participants were able to use the concept on the defined operation correctly, providing some mild evidence that this feature is useful and would be helpful to users of the query system. In fact, none of these participants made any errors with defined operations; their main queries and defined operation always "fit together" properly.

I. ACCURACY OF QUERY ONLY, NOT INCLUDING ACCURACY OF DEFINED OPERATION MODELS

	Subject					
	W	D	J	H	T	C
Correct				3(3)	2(2)	3(1)
Minor Syntax Errors						
Minor Conceptual Errors				1	2(1)	1
Moderate Errors						
Severe Errors	2	1	1			
Incomplete, but defined operation formulated						

II. ACCURACY OF ENTIRE QUERY (INCLUDES ERRORS IN DEFINED OPERATIONS)

	Subject					
	W	D	J	H	T	C
Correct						2
Minor Syntax Errors				2(2)	2(2)	
Minor Conceptual Errors				2(1)		2(1)
Moderate Errors					2(1)	
Severe Errors	2	1	1			
Incomplete, but defined operation formulated				2(2)		

III. ACCURACY OF DEFINED OPERATIONS ONLY

	Subject					
	W	D	J	H	T	C
Correct					1	
Minor Syntax Errors				2		
Minor Conceptual Errors				2		1
Moderate Errors						
Severe Errors						
Incomplete, but defined operation formulated						

III. CONCLUSIONS ON FEASILIBTY OF A DATA-BASE QUERY LANGUAGE BASED ON HOS SYSTEMS METHODOLOGY

The theoretical soundness of the idea of a data-base query language (or more generally, data manipulation language) is clearly demonstrated:

1) the addition of quantifiers to the syntax of HOS systems language makes the proof of query completeness elementary

2) The functional data model which is employed in our extension of the methodology is superior to other data models in being completely independent of implementation considerations

3) The axiomatic approach to data-types transfers readily to specifying integrity constraints of a data-base

4) The use of defined operations in HOS corresponds in a data-base context to the notion of user view or external view of the data-base

We have shown also that the technology is available among the HOS software tools to support the data-base query extensions.

The graphics editor commands turned out to be quite easy to learn, according to the experimenter's observation.

The results of the formal experiment regarding use of defined operations are inconclusive. Nevertheless, we will venture an interpretation of the results based on internal observation, in regard to use of the system.

1) The sharp division of the participants into two groups:

    a. those who were completely lost

    b. those who formulated correct queries, albeit a small number,

suggests that there is definitely a group of clerical personnel who will not profit from the integration of querying capabilities with general program specification capabilities, under the HOS methodology, and that there may be another group which has more analytic skills and can perform creative specification of queries and defined operations. (It should be stressed that none of the participants had computer backgrounds or even mathematics or science backgrounds. Group (b) included two college graduates, one in German, and one second year, business administration undergraduate. So, the system does allow people with a general educational background to specify queries.)

2) The system tested will be useful to managers and higher level staff and clerical workers in personalizing their access to their data-bases. To allow a broader range of personnel to access a data-base including those say, represented by (a) above, higher level personnel can design and implement standardized queries to be used by lower level personnel.

REFERENCES

1. M. Hamilton and S. Zeldin, "Higher Order Software - A Methodology for Defining Software," IEEE Transactions on Software Engineering, Vol. SE-2, No. 1, March 1976.

2. M. Hamilton and S. Zeldin, Axes Syntax Description, Technical Report No. 4, Higher Order Software, Inc., Cambridge, MA 1976.

3. Higher Order Software, Inc., USE.IT Reference Manual, Higher Order Software, Inc., Cambridge, MA, May 1982.

4. Higher Order Software, Inc., "Computable IDEF (C-IDEF): A Major Step Towards Increasing Productivity," Integrated Decision Support System (IDSS) Final Report for Period October 1978 - April 1981, Appendix I, Higher Order Software, Inc., Cambridge, MA, May 1981.

5. P. Madsen, "USE.IT Graphics," Internal Memo, Higher Order Software, Inc., Cambridge, MA, June 1981.

6. David W. Shipman, "The Functional Data Model and the Data Language DAPLEX," ACM Transactions on Database Systems, Vol. G, No. 1, March, 1981, Pages 140-173.

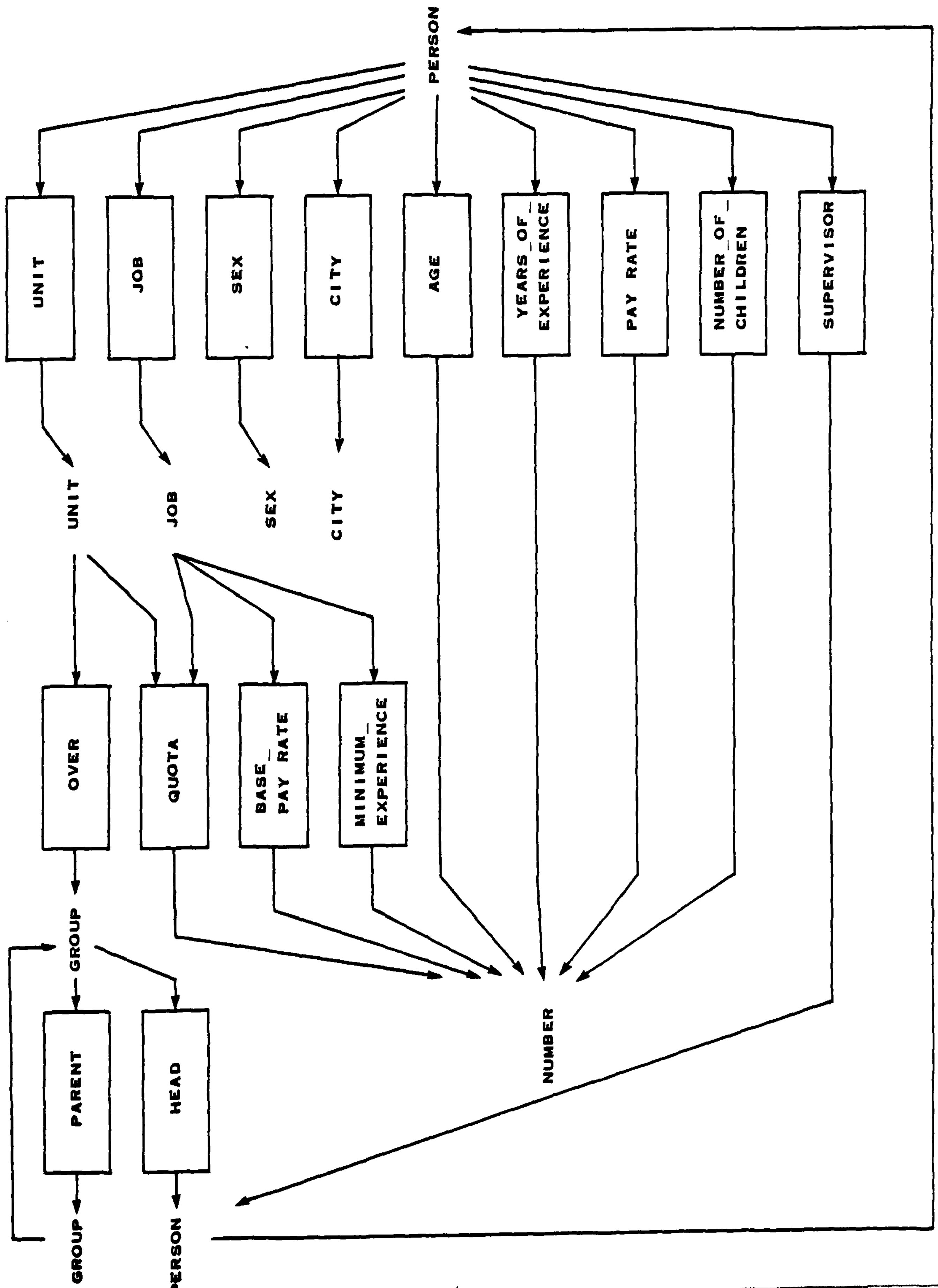
7. Jeffrey D. Ullman, Principles of Database Systems, Computer Science Press, Potomoc, MD 1980.

8. M.M. Zloof, "Query-by-Example: A Data Base Language," IBM Systems Journal, 16:4, pp. 324-343, 1977.

APPENDIX

USER REFERENCE HANDOUT

(INCLUDING TASK QUERIES)


PERSON
UNIT
JOB
SEX
CITY
AGE
YEARS_OF_EXPERIENCE
PAY RATE
NUMBER_OF_CHILDREN
SUPERVISOR
UNIT
JOB
SEX
CITY
OVER
QUOTA
BASE_PAY RATE
MINIMUM_EXPERIENCE
GROUP
NUMBER
PARENT
HEAD
GROUP
ERSON

EXAMPLES OF TYPES OF OBJECTS

PERSON:

Marilyn Brewer
Fred Smith

JOB:

Clerk Typist
Branch Head

UNIT:

Accounting Section
Budget Policy Section

GROUP:

Finance Branch
Research and Technology Division

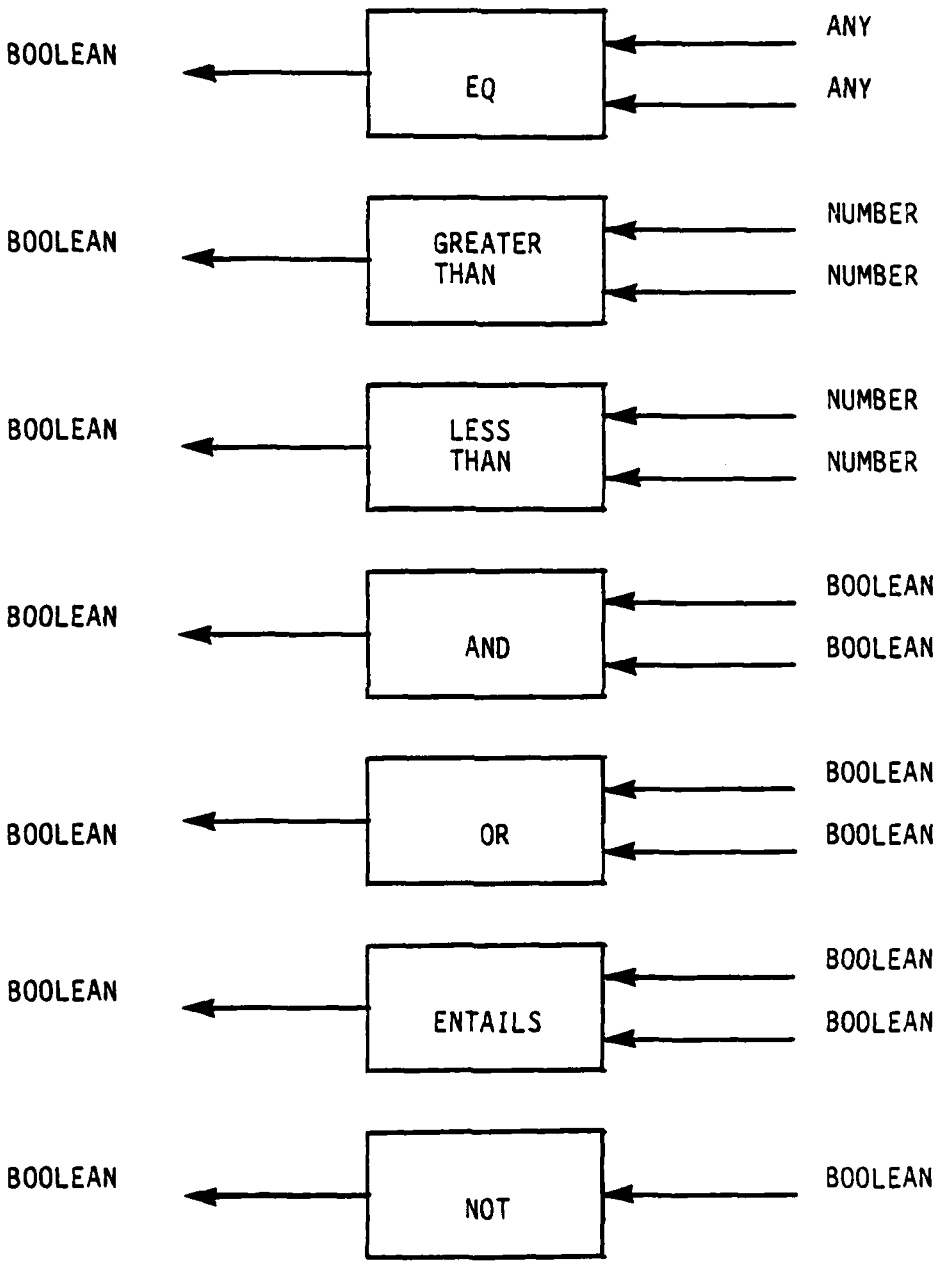
CITY:

Concord

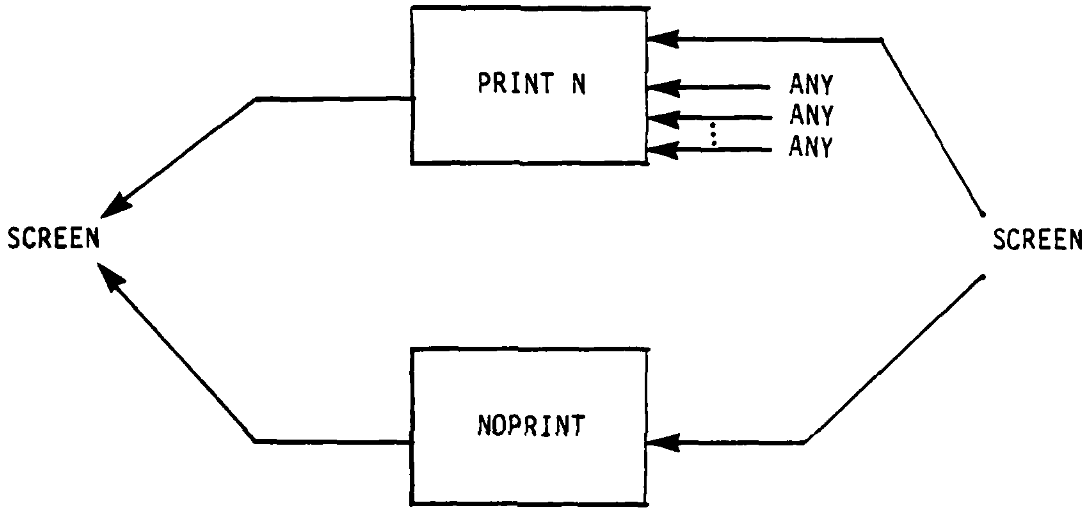
SEX:

Male
Female

GENERAL FUNCTIONS



PRINT FUNCTIONS



PRACTICE QUERIES

P1. How many children does MARILYN BREWER have?

P2. What is the BASE_PAY_RATE for a CLERK TYPIST?

P3. Please give me a list of the PERSON(s) who are BRANCH HEAD(s).

P4. How old is the HEAD of the ACCOUNTING SECTION?

P5. Give me the NUMBER_OF_CHILDREN and CITY of residence for ETHYL MULLINS.

P6. I'm writing a letter to ROBIN FARROW, and I need to know whether Robin is a man or woman (MALE or FEMALE).

P7. Here's a request from the County. They want to know about only children, so I need a list of the PERSON(s) with exactly one child (NUMBER_OF_ CHILDREN is one), and also give the person's JOB.

P8. For my own use, give me a list of the PERSON(s) with exactly one child (NUMBER_OF_CHILDREN is one), and also give the person's JOB.

P9. The County wants to interview parents of only children. I need a list of the PERSON(s) with exactly one child (NUMBER_OF_CHILDREN is one), and also their SUPERVISOR, so I can tell them to authorize time off from work.

ACTUAL QUERIES I

1. What PERSON is the HEAD of the FINANCE BRANCH?

2. I'm considering a readjustment of requirements for some of our lower-grade JOB(s). Give me a list of JOB(s) with a BASE_PAY_RATE of 12000 or less.

3. I need a list of all the PERSON(s) we have who are FEMALE.

4. I'd like the JOB(s) with BASE_PAY_RATE of 12000 or less and also the MINIMUM_EXPERIENCE for each.

5. I need a list of the PERSON(s) who are BRANCH HEAD(s).

6. I need to arrange a conference with our top-ranking women. Give me a list of the PERSON(s) who are FEMALE and whose PAY_RATE is over 25000.

7. Now I need a list of the PERSON(s) and their SEX who have JOB(s) for which the BASE_ PAY_RATE is 12000 or less.

8. I need another listing for those lower-grade JOB(s). Can you give me a list of PERSON(s) and their AGE for everyone who holds a job with a BASE_PAY_RATE of 12000 or less.

9. The people who send out city tax information need a list of every CITY where we have personnel residing.

10. Some of the women in the organzation have expressed interest in a meeting for working mothers. To help them out, I want to give them a list of the PERSON(s) who are FEMALE, what CITY they live in, and the NUMBER_OF_CHILDREN they have.

11. What is the AGE of FRED SMITH?

12. There is a women's caucus of the Association for Computer Programmers that wants the PERSON(s) who are FEMALE and who work as PROGRAMMERS. Please get me that list.

ACTUAL QUERIES II

13. I've become concerned lately about people whose YEARS_OF_EXPERIENCE don't meet the MINIMUM_EXPERIENCE for their JOB(s). I'd like a list of such PERSON(s) and their JOB(s).

14. For everyone whose YEARS_OF_EXPERIENCE don't meet the MINIMUM_EXPERIENCE for their JOB, I want to know their SUPERVISOR and the SUPERVISOR(s) AGE.

15. I need to know what PERSON is the HEAD of the BUDGET SECTION before I meet him or her.

16. What PERSON(s) have a higher PAY_RATE than BRIAN WEISBERGER?

17. I need a separate list of PERSON(s) whose YEARS_OF_EXPERIENCE don't meet the MIMIMUN_EXPERIENCE for their JOB, who also live in CONCORD.

18. I have forgotten what the JOB that GEORGE SELLING has is called, and I don't particularly care, but I need a list of the PERSON(s) with that JOB.

19. I also need to know what PERSON is the HEAD of the RESEARCH AND TECHNOLOGY DIVISION. I also need to know his or her AGE, SEX, and NUMBER_OF_CHILDREN.

20. Which of those PERSON(s) whose YEARS_OF_EXPERIENCE don't meet the MINIMUM_EXPERIENCE for their JOB are MALE?

21. Can you give me a list of all the PERSON(s) who are CLERK TYPIST(s)?

22. Provide a list of PERSON(s) who live in DOVER and give me their AGE, SEX, and NUMBER_OF_CHILDREN. I want to arrange a public realations effort in that community, and need to select some contacts there.

23. I need a list of PERSON(s) whose PAY_RATE is over 25000, along with AGE, SEX, and NUMBER_OF_CHILDREN. I'm looking at adjustments in that upper range.

24. What is the BASE_PAY_RATE for the HEAD of the BUDGET POLICY SECTION?

ACTUAL QUERIES III

25. Our outside consultants are doing an evaluation of our typing practices. I need a list of UNIT(s) that have some CLERK TYPIST(s).

26. The outside consultants also wish to cut back the total number of personnel. To do this, they wish to identify personnel who hold "duplicated jobs". Duplicated jobs are those in a UNIT for which the UNIT has a QUOTA of more than one (i.e., since the Budget Policy Section has a quota of two clerk typists, clerk typist is a duplicated job). I need a list of PERSON(s) holding duplicated JOB(s) and the CITY they live in.

27. I want a list of the PERSON(s) who are HEAD(s) of GROUP(s) and who have no children (NUMBER_OF_CHILDREN is zero).

28. About duplicated jobs--I need a list of the PERSON(s) who hold them, their JOB, and the UNIT they work in.

29. The outside-consultants now want a list of the PERSON(s) who are HEAD(s) of GROUP(s) whose UNIT(s) have some CLERK TYPIST(s). They also want to know the PAY_RATE(s) of those PERSON(s).

30. I have some extra money to disperse. Give me a list of the PERSON(S) whose PAY_RATE is less than 12000 and who also have more than two children (NUMBER_OF_CHILDREN is greater than two).

31. I need an extra CLERK TYPIST to help out in the Data Processing Section. Give me a list of the PERSON(s) who are CLERK TYPIST(s) who work in the BUDGET POLICY SECTION and who have at least two YEARS_OF_EXPERIENCE.

32. For those UNIT(s) with some CLERK TYPIST(s), the consultants need a list of the PARENT(s) of GROUP(s) which are OVER the UNITS with CLERK TYPIST(s).

33. The consultants want to talk to the SUPERVISOR of PERSON(s) who hold duplicated JOB(s). Give me a list of those SUPERVISOR(s).

34. I need a list of PERSON(s) who hold duplicated JOB(s) and whose PAY_RATE is more than the BASE_PAY_RATE for their JOB.

35. I have to satisfy affirmative action requirements. Do we still have any UNIT(s) which have only men (MALE)? I need a list of those UNIT(s).

36. For each UNIT that actually has a CLERK TYPIST, the consultants need to know its QUOTA of CLERK TYPIST(s).

DISTRIBUTION LIST

OFFICE OF NAVAL RESEARCH

Code 442

TECHNICAL REPORTS DISTRIBUTION LIST

Procurement Contracting Officer
Office of Naval Research
Code 614A: CEA
800 North Quincy Street
Arlington, VA 22217

Office of Naval Research
Code 442EP
800 North Quincy Street
Arlington, VA 22217 (2 copies)

Defense Technical Information Center
Cameron Station, Building 5
Alexandria, VA 22314 (12 copies)

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, DC 20375 (6 copies)

Commanding Officer, ONREAST
Barnes Building
495 Summer Street
Boston, MA 02210

OSD

Capt. Paul R. Chatelier
Office of the Deputy Under Secretary
of Defense
OUSDRE (E & LS)
Pentagon, Room 3D129
Washington, DC 203101

Dr. Stuart Starr
Office of the Assistant Secretary
of Defense (C^3I)
Pentagon, Room 3E182
Washington, DC 203101

Department of the Navy

Dr. Robert G. Smith
Office of the Chief of Naval
Operations OP987H
Personnel Logistics Plans
Washington, DC 20350

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, DC 20380

Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Communication & Computer
Technology Programs
Code 240
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Manpower, Personnel and Training
Programs
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Information Systems Programs
Code 433
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Phillip J. Andrews
Naval Sea Systems Command
NAVSEA 03-1
Washington, DC 20362

Dr. Arthur Bachrach
Naval Medical Research Institute
Behavioral Sciences Department
Bethesda, MD 20014

Dr. Robert Blanchard
Naval Personnel Research and
Development Center
Command and Support Systems
San Diego, CA 92152

Dean of Research Administration
Naval Postgraduate School
Monterey, CA 93940

Warren Lewis
Naval Ocean Systems Center
Human Engineering Branch
Code 8231
San Diego, CA 92152

Dr. Thomas McAndrew
Naval Undersea Systems Center
Code 32
New London, CT 06320

Dr. James McGrath
CINCLANT FLT HQS
Code 04E1
Norfolk, VA 23511

Dr. George Moeller
USN Submarine Medical Research Lab
Human Factors Engineering Branch
Naval Submarine Base
New London, CT 06340

Commander
Naval Air Systems Command
Human Factors Programs
NAVAIR 340F
Washington, DC 20361

Commander
Naval Air Systems Command
Crew Station Design
NAVAIR 5313
Washington, DC 20361

Naval Air Test Center
Systems Engineering Test Directorate
Human Factors Section
Patuxent River, MD 20670

Commander
Naval Electronics Systems Command
Code 81323
NC #1 Room 4E56
Washington, DC 20360

Dr. Eric Gunderson
Naval Health Research Center
San Diego, CA 92152

Naval Material Command
NAVMAT 0722 Room 508
800 North Quincy Street
Arlington, VA 22217

Naval Training Equipment Center
Human Factors Department
Code N-71
Orlando, FL 32813

Naval Training Equipment Center
Attn: Technical Library
Orlando, FL 32813

Larry Olmstead
Code N-32
Naval Surface Weapons Center/DL
Dahlgren, VA 22448

Dr. Gary Poock
Naval Postgraduate School
Operations Research Department
Monterey, CA 93940

H. Talkington
Naval Ocean Systems Center
Ocean Engineering Department
San Diego, CA 92152

Walter P. Warner
Code K02
Strategic Systems Department
Naval Surface Weapons Center
Dahlgren, VA 22448

<u>Department of the Army</u>

J. Barber
Headquarters, Department of the Army
DAPE-MBR
Washington, DC 20310

Technical Director
U.S. Army Human Engineering Lab
Aberdeen Proving Ground, MD 21005

Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Director, Organizations and Systems
Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

<u>Department of the Air Force</u>

Air Force Office of Scientific Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, DC 20332

Chief, Systems Engineering Branch
Human Engineering Division
AMRL/HES
Wright-Patterson AFB, OH 45433

<u>Other Organizations</u>

Dr. Craig Fields
Director, System Science Office
DARPA
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Robert T. Hennessy
NAS-National Research Council
Committee on Human Factors
2101 Constitution Avenue, N.W.
Washington, DC 20418

Dr. Lloyd Hitchcock
Federal Aviation Administration
ACT 200
Atlantic City Airport, NJ 08405

Dr. M. Montemerlo
NASA Headquarters
Human Factors & Simulation RTE-6
Washington, DC 20546