GRAPHICAL USER LANGUAGES FOR QUERYING INFORMATION: WHERE TO LOOK FOR CRITERIA?

Gabriele Rohr
Science Center Heidelberg, IBM Germany
Tiergartenstr. 15
D-6900 Heidelberg

ABSTRACT
The different parameters of a graphical user language in general, concerning kinds of representation and kinds of operations on these representations will be outlined. These parameters will be related to the characteristics of tasks which they more or less support. Special aspects of a graphical query language have been derived from psychological findings. The conceptual framework from Jackendoff [1] will be used to model user concepts in graphical query languages and to translate these concepts to relational database semantics.

1. WHAT IS A GRAPHICAL USER LANGUAGE?
Graphical user languages can be generally regarded as what is usually called direct manipulation. A clear definition of what direct manipulation will mean is needed here hence there is still a lot of confusion about what it is and what it is not.

First of all, direct manipulation must be defined as working on graphical objects, i.e., also in text editing, text is regarded as a graphical object whose graphical properties (font, size, thickness, place, etc.) can be directly and visibly changed. Generally, graphical objects and their arrangement in space are supposed to express system components (objects) and their relations (space). The graphical representation can be more abstract showing mainly spatial relations (see fig. 1a) or more concrete (picture like) using necessarily metaphors to describe the semantics of the system's components (see fig. 1b), and a combination of the two (see fig.1c). Consequently, one of the most important problems is to find the adequate metaphor expressing clearly the semantics of these objects.

Operations are defined by pointing at these objects arranged in space or at locations of a spatial structure and initiating an action by a special clicking event. Objects or entities pointed at and clicked will then change parameters as color and shape or may be 'dragged' around which means the implementation of some kind of animation. Those graphical 'events' have to be clearly and consistently related to system's operation semantics.

Because nearly everything will take place in space and on objects it is important to make all functional relevant components of the software system visible. Furthermore, all changes of the system state have to be seen immediately in a graphical way (direct event). This is opposite to control statements which initiate a chain of events where only the final result might be visible.

The conceptual requirements are still a concern of research. In order not to be more confusing than helpful the semantics of graphical representations and concepts at the user's side have to be clearly and unambiguously mapped to the semantics of the software system. This means that a graphical user language has to be developed, i.e., the semantics of the interface has to be formally defined by means of an adequate specification language. Furthermore, it requires also a great deal of detailed investigations on users' understanding of graphical concepts.

Before deciding to introduce a graphical user language instead of other kinds of user languages the characteristics of the tasks that have to be performed with a system, and the characteristics of the single operations within the tasks have to be analysed.
Three global areas of task characteristics can be distinguished: 1. different views on different parts of an overall structure, 2. the control of sequences of operations, 3. the allocation or relocation of data sets to distinct classes and their computation. The second area, control of sequences, can be furthermore split up into two aspects: 1. the control of a linear sequence of operations, and 2. the reconstruction of a sequence from randomly ordered elements. Gersendörfer and Rohr [2] found out that independent of the individually preferred problem solving strategy (visualising, verbalising, formalising) tasks requiring different views on an overall structure were best performed when represented by means of a picture, and tasks requiring allocating and relocating data sets to distinct classes and conducting computations on them within and between classes were best performed when represented by means of a formal table. The reconstruction of operation sequences was first assumed by them to go best with natural language (text) but came out as the worst condition at all and going best with pictures for formalisers and with formal tables for visualisers. This seems to support a more formal graphics representation to be best for this kind of task characteristics. With a later experiment they found out that natural language was going best with the control of linear operation sequences.

While the task characteristics determine in some way globally the best representation strategy the operation characteristics determine how this is worked out in the detail. There are two general kinds of operations: spatial operations and existential operations [3]. Spatial operations are defined by the existence of an object which is moved in space. Spatial operations have still to be worked out. The main operation is a spatial operation and best represented by direct manipulation. Existential operations are defined by state transformations of objects and the formulation of their condition. These kinds of operations will be best performed when formulated as words.

When deciding for a graphical user language these aspects have to be regarded. An application which main task characteristic consists of views on structures and which main operations are spatial operations is best suited by a graphical user interface with direct manipulation in contrary to one which main task characteristic is the control of linear operation sequences and which main operations are existential ones.

2. GLOBAL LEVELS OF SEARCH

Querying information can be regarded as stepwise delimiting the search space: of what do I want to have information, what information structure, for which purpose. This applies especially for databases. In databases we have to distinguish three operation levels which have to be analysed separately with respect to task and operation characteristics. The first level is the selection of parts of a database to be operated on. The second level is concerned with defining a relation within the part, aiming at a special result (query). The third level is concerned with operations on the query result.

The first level can be regarded as a kind of zoom on the database, i.e. a selected view. The main operation has a spatial characteristic, i.e. take parts from a whole structure. Consequently the first level will be best suited by a concrete metaphorical graphical representation (because of orientation support: people know then where to look) with a direct selection of objects and their attributes (pop-up menus in objects: mixed graphics). In the second level the view on the structural relations and dependencies of the data sets becomes important. The main operations are: creating different views on these relations according to the intended query result which is a spatial operation (i.e. restructuring in space), and defining conditions for the view of the final result which is an existential operation. Graphics is also here the best overall representation form showing the structure of the data. Direct manipulation must allow to show the structural dependencies from different chosen points, where it must be possible to enter conditions into the structure by means of words. The third level is mainly determined by the purpose of the result. Either it means again analysing structured relations, consequently, it has to be represented graphically, or it means the computation of the elements. Under this condition a formal table format of the data will be most suitable. The operations can be different. One operation might be to shift the elements around which is a spatial operation and best represented by direct manipulation of table elements. The more frequent operation will be to condense sets of elements, i.e. compute them, which is an existential operation. This could be done by entering the function in the respective table field (defining conditions and transformations at a certain place) which is a kind of a mixed concept. This mixed concept we will find in spreadsheet operations and in QBE, which will be discussed later on.

As could be shown, it is very important to analyse an application very carefully before deciding where to introduce direct manipulation. The abstract graphical representation supporting the conceptual structure in querying has still to be worked out.

3. PSYCHOLOGICAL CONCEPTS FOR SPATIAL REPRESENTATIONS IN QUESTIONING

To make the conceptual structure behind a question somehow clear, let us take an example. Let us assume we became acquainted with someone named Miller and learned from him that he is a project leader. Now, we know that a project is defined by a special working area, and has a certain number of employees working in it and reporting to the project leader. If we wanted to ask someone about the kind of project Miller leads and the people working for him we would rather ask...
"Can you tell me of Miller, the project leader, what kind of project does he lead, and who is working for him?"

"Who are the employees of the project where the project leader is named Miller and what is the description of the project?"

where the second question is supposed to be in elaborated language, but harder to understand. What we really ask for are the values of attributes of Miller we know he must have. The grammatical structure of the first question expression reflects more a conceptual structure in a top-down procedure than the second question expression. The assumed conceptual structure of the question is shown in fig. 2.

**Figure 2: Query structure: only values of Known structural elements can be asked for**

Easy formulation in elaborated natural language is true for very simple questions where only a list of attributes of one object is asked for (e.g., ‘what is the color, shape, and price of the shirt?’). However, if attributes of different objects in a defined dependency are asked for in combination something unexpected happens: the grammar used does not follow any more the usual grammatical rules. Zoeppritz [4] investigated these effects at natural language database interfaces. As a prominent example the following query of one of the users can serve:

"Which German grade in seventh class has how many pupils?"

What has happened? It seems that the correct formulation of the question How many pupils of the seventh class have which grades in German? does not meet the underlying question concept. Otherwise, nobody would ask like this in colloquial speech. Here, it is more likely to ask: In the German course in the seventh class, how is the distribution of grades? (Implicit knowledge: distribution, i.e. how many pupils are related to each possible grade?)

The problem with the 'strange' question expression is, that there is probably no usual grammatical structure to map the top-down procedure on the given conceptual structure. This conceptual structure seems to be important in formulating queries, and if it becomes complex it cannot be easily mapped to a correct but differing grammatical structure.

Investigations of Shneiderman [5] seem to support this hypothesis. The authors found out that it was easier for users to express their questions in SEQUEL correctly, if they were complicated, than in natural language. Even if SEQUEL does not correspond to the conceptual structure it has a fixed query structure while natural language includes the additional effort to find the right grammatical structure out of a number of possible variants.

To support question asking means now, to find out by which rules conceptual structures of queries are build up, and then to directly support them.

The most detailed investigations on conceptual structures were done for three-rank order problems like A > B, B > C and the question upon this structure ‘which is the greatest/smallest?’ (for an overview see [6]). Three main parameters were found out to be important in formulating the structural relationships between A, B, and C with respect to the ease of answering the question stated above:

1. The existence of a conceptual end-anchor: 
   A > B, C < B where B is the reference;
2. the correspondence of logical and grammatical subject: 
   B > C, B < A , which means B is between A and C , and B is logical subject;
3. the linearity of the presented order: 
   A > B, B > C .

If any of these parameters is found in an expression describing three-rank order structures, questions for the greatest and the smallest element can be answered more quickly and easily.

In higher-rank order problems the third parameter (linearity) becomes dominant because there is no longer one fixed logical subject or one reference possible. According to Kosslyn [7] it is not so easy to answer the question

**Is Betty older than Frank?**

if the overall order is presented in the form

1. Michael is older than Betty,
2. John is younger than Betty,
3. John is older than Frank,
4. Susan is younger than Frank.

than if it is presented as
1. Michael is older than Betty,
2. Betty is older than John,
3. John is older than Frank,
4. Frank is older than Susan.

The easiest way, however, to answer any of those questions independent of the direction (older/younger) is given if the order is presented graphically (see fig. 3). Here, the conceptual structure has not to be build up but can be seen immediately. This leads to the assumption that, in some way, the graphical representation maps best the representation of the conceptual structure operated on in question asking.

Figure 3: Best representation to answer "who is older/younger" questions

If we look at the parameters supporting the solution of three-rank order problems, the conceptual end-anchor and the correspondence of logical and grammatical subject have also, like the linearity of the order, something to do with a spatial concept. The logical subject can be considered as object of which its place is referred by the places of objects A and C. In reverse, the end-anchor can be considered as reference place of object B for the objects A and C (see fig. 4). According to the formal descriptions of mental semantics of Jackendoff [1] this is a basic description of a place-function, like

\[
\text{<THING B> PLACEFUNCTION <PLACE THING A> PLACEFUNCTION <PLACE THING C>}
\]

or

\[
\text{<THING A> PLACEFUNCTION <THING C> PLACEFUNCTION <PLACE THING B>}
\]

or

\[
\text{<THING C> PLACEFUNCTION <PLACE THING B> PLACEFUNCTION <PLACE THING A>}
\]

Possession functions express relations between things and their attributes, attributes belonging to things (Jackendoff 1983). Consequently, with the possession function conceptual dependencies in databases can be expressed metaphorically in the human's mind: thing A has (possess) thing B, thing B belongs (is possessed by) to thing A. This can also be a metaphor to express dependencies in databases. Employees belong to a project leader, and employees possess a salary. If the project leader is the fixed point (known subject: Miller) to determine the employee names and their salary which will be asked for, the question arises how the question will most easily be structured.

- With the conceptual end-anchor: Miller possess which employees, which salary belongs to these employees?
- with the correspondence of logical/grammatical subject: Which employees have Miller as leader and possess which salary?
- with the linearity of the possession function: Miller possess which employees having which salary?

This analysis is important when defining the semantics of a special query view on the graphical database structure (i.e., which thing has to be placed on the top to express what?). E.g., if the third parameter holds in question asking, "Miller" has to be placed on top, otherwise for the second parameter it will be "employees".
Another question is, how people structure their knowledge space they ask on. Do they see object entities with attributes and relate the objects over the attributes, or do they first create an object in mind with specified attributes they ask for? Let us take as an example the question of Miller and his employees with their salary. In the first case the object "project" with its specified attribute "leader" will be related to the object "employee" with its attributes "name" and "salary". In the second case, the defined object "Miller" as a project leader will be asked for its attributes "employee names" with their attributes "salary". These different mental operations have different implications for their graphical representation when question asking should be aided by graphics. These implications we try to specify and relate to graphical user languages for database queries.

4. SPECIFICATION OF THE QUERY SPACE

The first thing to be done with the query space is to determine what are the things, the attributes they possess, how a token of a thing is defined, and how it is exemplified, what is an instance of what, and which things are possessed by other things. This has then to be related explicitly to a set of place-functions: where are things possessed by a thing, where are tokens of a type, etc. By means of this procedure we will get a representation of the query - "world". In the same way we have to proceed with the operations possible within this "world". To determine the possible operations means to look for what can be changed in the overall structure of the "world": what can be done with the possession of things, can they be viewed reverse, i.e. as belong-to-relations which will mean changing the view, and consequently, the place of the things within the representation, can tokens of things be specified by means of one attribute, etc.

The query space we have to specify in order to formulate a graphical user language is strongly dependent on the data definition and manipulation form of the underlying database structure. Because of its flexibility we rely on a relational DB-system.

In the relational data system we find data defined in entities with attributes of which a value-tuple is associated with each individual of the entity and where at least one attribute characterizes each individual of the entity (candidate key). We also find entities with attributes which define the relation between entities (foreign keys).

Specifying entities according to Jackendoff's [1] formalism they can be regarded as type-things where each token of this type is an instance of this type-thing, possessing a set of properties with characteristic values for each token. We then can formulate a set of rules describing these facts:

(4.1) (entity-attribute relation)

<THING TYPE A> POSSESS <THING TYPE PROPERTY>

(4.2) (candidate key)

<THING TYPE A> IS_EXEMPLIFIED_BY <THING TOKEN OF PROP. IDENT>

(4.3) (tuples)

<THING TOKEN ai> IS_INSTANCE_OF <THING TYPE A>

(4.4) (cardinality of tuple)

<THING TOKEN ai> POSSESS <THING TOKEN OF PROPERTY>

These rules can be translated to place-functions according to (3.1) and (3.2) except rule (4.2):

(4.1.1)

<THING TYPE PROPERTY> PLACEFUNCTION(on) <THING TYPE A>

(4.3.1)

<THING TOKEN ai> PLACEFUNCTION(in) <THING TYPE A>

(4.4.1)

<THING TOKEN of PROPERTY> PLACEFUNCTION(on) <THING TOKEN ai>

For (4.2) we define tokens to be on each other within the type space:

(4.2.1)

<TOKEN ai> PLACEFUNCTION(on) <TOKEN aj> PLACEFUNCTION(in) <TYPE A>

The next thing to define is the relation between entities. Hence this is done in relational DBs by means of the identifier attribute, we specify: token of type-thing A possess type-thing B. This can be formulated as:

(4.5) (foreign key relations)

<TOKEN ai> POSSESS <THING TYPE B>

and translated to a place-function expression as:

(4.5.1)

<THING TYPE B> PLACEFUNCTION(on) <THING TOKEN ai>
QBE is an already existing database query frame for relational DBs which is supposed to have some graphical language aspects ([8]). It could be really regarded so if some more direct manipulation operations were installed. QBE assumes a somewhat more restricted graphical user concept: attributes are on things and certain things have certain attributes in common (not spatial expression of entity-relationships). The question is how users' query concepts will be influenced by this graphical concept. Therefore, we must try to specify its rules of place-functions represented implicitly in its tabular graphics. The following rules can be derived from analysing the QBE representation of the query space:

\[(4.1.2)\]
\[
\text{\textless TYPE PROPERTY\textgreater \ PLACEFUNCTION(on) \ \textless \ THING TYPE A\textgreater}
\]

\[(4.2.2)\]
\[
\text{\textless TOKEN ai\textgreater \ PLACEFUNCTION(left,on) \ \textless \ THING TYPE A\textgreater \ \PLACEFUNCTION(under) \ \textless \ THING TYPE PROPERTY IDENT\textgreater}
\]

\[(4.3.2)\]
\[
\text{\textless TOKEN ai\textgreater \ PLACEFUNCTION(under) \ \textless \ THING TYPE A\textgreater}
\]

\[(4.4.2)\]
\[
\text{\textless TOKEN of PROPERTY\textgreater \ PLACEFUNCTION(on) \ \textless \ THING TOKEN ai\textgreater}
\]

\[(4.5.2)\]
\[
\text{\textless PROPERTY\textgreater \ PLACEFUNCTION(on) \ \textless \ THING TYPE A\textgreater \ \textless \ THING TYPE B\textgreater}
\]

By means of the rules stated above it is possible, given a database schema as shown in fig. 6, to generate a graphical interface of the query space which can be directly operated on (see fig. 7a and 7b).

![Figure 5: Graphical representation of elements and place-rules for GUL-Q (Graphical User Language - Query: own approach) and QBE.](image)

![Figure 6: Database schema for a small department database](image)

In a next step all questions/operations possible on this structure have to be specified and transformed into place operations. Generally speaking, querying / operating on this query space means defining views on token-properties dependencies, i.e., to make the structure show:

1. selected properties of all tokens of a type,
2. all tokens of a defined property,
3. property values of a defined token,
4. things possessed by tokens directly defined, or defined over properties,
5. etc. (all possible combinations of (1)-(4)).

In relational DBs these conceptual user semantics correspond to the following DB-semantics:

1. projection,
2. selection,
3. selection + projection,
4. join.

It is not as difficult to see that these representation rules are quite different from those we deduced by means of the Jackendoff formalism, and that they are leading to another conceptual structure. Fig. 5 shows the visual realisations of both sets of rules.
Hence these DB-semantics have a well defined algorithm for retrieving data, the translation from the graphical direct manipulation language to the DB language can be done at this level.

Summarizing, we have some few semantically different basic operations of which a combination can be allocated to different queries:

1. showing the dependencies from one/some property defined token
   \[
   \text{PUT} \left( \langle \text{THING TOKEN} \rangle \text{ PLACEFUNCTION(on)} \langle \text{PLACE BASIC} \rangle \right)
   \]
   where PLACE BASIC is the one place where all other THINGS are on

2. selecting properties of tokens to be shown
   \[
   \text{SELECT} \left( \langle \text{THING PROPERTY} \rangle \text{ PATH(from)} \langle \text{THING TYPE} \rangle \right)
   \]

3. selecting the tokens of a type
   \[
   \text{SELECT} \left( \langle \text{THING TOKEN} \rangle \text{ PATH(from)} \langle \text{THING TYPE} \rangle \right)
   \]

4. defining the token of a property to define the tokens of a type selected
   \[
   \text{DEFINE} \left( \langle \text{THING PROPERTY} \rangle \text{ PLACEFUNCTION(on)} \langle \text{THING TOKEN} \rangle \right)
   \]

This all can be done directly on the displayed structure. With a pointing device (e.g., a mouse) one can point at a property of a type-thing and click and it will be highlighted. Pointing and clicking at a token field will wait for an entry in the field defining the token (existential operation: word input). Pointing and clicking with a special button on a property field will put this field on the basic place making it to the identifier of the type-thing. Pointing and clicking at the tokens of a type-thing will highlight the tokens indicating to be selected. With the rearranged structure with its shown dependencies and highlighted and defined areas the user can now check his/her question.

The result of such a query process for the question we formulated in section 3 (see also fig. 2) is shown in fig. 8. The structure build up in this representation resembles the conceptual structure assumed shown in fig. 2, except that here is more information about token dependencies available.

The same procedure for specifying operations on the displayed data structure has been done for the QBE version. Also here, we have areas which can be pointed at and clicked (properties / type-things) and token fields where conditions can be entered. However, more has to be defined in the token fields, and dependencies cannot be shown directly. See the same query result for the QBE version in fig. 9.

By means of our specification rules we are now also able to describe the first operating level in search, i.e., selecting parts of the query space to be operated on (see section 2). Each type-thing will be represented by a concrete icon and arranged in space according to the possession function between the type-things, all type-things together building up some kind of a supericon with selectable elements, selected by pointing and clicking. For a rough selection of properties which are important for further questions each type-thing will show a pop-up menu when clicked, similar
to the representation shown in fig. 1c. The operation space at this search level is clearly defined in its stronger limitation: it is only possible to select type-things and their properties.

Figure 9: The resulting query formulation for the QBE-like representation

5. TESTING GRAPHICAL LANGUAGE CONCEPTS: USER EXPERIMENTS

The conceptual implications of the different graphical languages have to be tested out with users, especially the implications for different query structures. One important difference between the QBE-like and the GUL-Q language is that GUL-Q shows directly type-token dependencies which could be relevant in nested queries like: In the German course in the 7th class give me for each grade the number of pupils” as discussed in section 3.

To learn more about mapping of graphical languages to users’ query concepts a strong specification of question structures in user experiments is needed but still missing: Varying systematically

- the degree of nestings (structural dependencies) within the query,
- the number of conditions formulated,
- the amount of “circular” conditions, i.e., the comparison of two attributes of two different entities (e.g., “where employee name = project leader”) where the QBE-like representation might do better.

With GUL-Q we define hierarchical views on structural dependencies. Most human concepts rely on hierarchical views, and they operate easier on them; but circular dependencies are sometimes necessary to be regarded. How to support them is a question of further research.

At the time we are running experiments on these questions with a prototype implementation of the graphical user semantics specification for GUL-Q and QBE-like variants, including user protocols of the query behavior. The different question aspects are tested systematically. The results hopefully, will give us some more insights.

REFERENCES