Iconic and Diagrammatic Interfaces: An Integrated Approach

T. Catarci, A. Massari and G. Santucci

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria, 113 - 00198 Roma

Abstract

In this paper we first analyze the motivations for using a visual approach to querying a database. Afterwards we consider the two main visual expressions, i.e. icons and diagrams, used to represent both data and processes, together with their main advantages and limitations. Based on these considerations, we describe a man-machine interface, together with a visual query language providing an integrated, both iconic and diagrammatic, approach.

1. Introduction

Many studies have been carried out to improve man-machine interaction by conveying information in a way directly graspable by a human, without large efforts nor an intermediate learning medium. In this sense, a new approach has been proposed based on visualization, i.e. on the representation of drawings/images which, when used as metaphors, bear information at an adequate level of abstraction and in a concise form. This visual approach is advantageous due to the reduction of the mental load and to the immediate availability of descriptions of the computation processes and their interrelationships. Based on this approach, programming environments, languages and systems have been developed in the last years highlighting some interesting properties which make such products friendly, easy to understand and use.

We call visual system any system providing a visual man-machine interaction. Such interaction is typically based on a visual language. In order to give a concise definition of visual language, we may say, according to the point of view of many authors (e.g. 3, 7), that it implies the systematic use of visual expressions (such as icons, drawings or gestures) to convey a meaning in a formal way. Frequently, people have used visual languages, iconic languages and graphical languages as synonyms. If we follow the above definition they all visual languages, but we prefer to call iconic languages only the languages using images and icons extensively, while graphical languages are those primarily using drawings (such as graphs, flow-charts, block-diagrams, etc.).

A visual language may handle, by means of visual constructs, objects that do not necessarily have a visual representation (e.g. traditional data types like arrays, lists, stacks, and application oriented data types like forms, documents, etc.). In order to obtain a friendly man-machine interface, the objects are presented visually; therefore, a visual representation is associated with them. The database field is one of the most promising application areas of visual languages for many reasons. One reason is that very often the database is queried by a casual user, who does not know a restricted natural language, both the dependency on the native language of the user and the limitations imposed by the application area are avoided. Finally, the typical process of query formulation itself encourages using visual languages. In fact, we can distinguish three steps: the first step selects those parts of the database we should operate on; the second step defines the relations within the part in order to produce the query result and the third step operates on the query result, if necessary. The first step is the best candidate for a metaphorical visual representation since it can be seen as a kind of zoom on the database in order to select the parts of interest. In the second step, the main operations are used to express conditions on the structural relations according to the formulated query. A visual representation is useful in order to show both the database contents and the query structure. Lastly, third step, it may be useful to visually represent the structured relations for analyzing the query result.

In Sections 2 and 3 graphical and iconic interfaces are described respectively. Section 4 deals with the basic ideas of the integrated approach. The system architecture is shown in Section 5 and its main modules are deepened in Section 6. Finally, conclusions are drawn in Section 7.

2. Graphical Interfaces

Graphics is generally an added value to existing programs where some objects and their relationships may be visualized, particularly in block diagrams representing programs, in diagrams representing state transitions or in trees showing hierarchical dependencies, etc. In all these cases graphical constructs are convenient for describing functional relationships. In fact, graphical systems convey meanings through a description of a multiply connected world of objects and their inter-relationships. Moreover, in the database world the most popular data model is presently the Entity-Relationship (E-R) model which is effectively and efficiently represented by graphical constructs. In graphical languages based on such a model, and more generally on semantic models, objects and relationships defined among them are conceptually represented by means of high level abstractions like aggregations and generalizations. In principle every object of a real world application is represented by means of a specific concept in the schema. Furthermore, every concept of the schema can be represented by means of a graphical symbol in a diagram. As a consequence, the user is released from syntactic and implementation details, and the query can be naturally expressed by pointing directly to objects and spatially navigating among them.

A typical system for querying databases, using diagrams as standard user interface, is Query By Diagram* (QBD* in the following) [1]. The system makes use of a conceptual data model, a query language on this model and a graphical user interface. The conceptual model is the Entity-Relationship Model [5], augmented by generalization abstractions [9]; the query language is mainly a navigational language on diagrams representing conceptual schemas. The user first interacts with the conceptual schema to understand its information content, and extracts the subschema of interest containing the concepts involved in the query, then he may express the query, defining all its procedural characteristics. During the "navigation" activity, the user selects all the concepts involved in the query: first a central concept, called main concept, is chosen, that can be seen as the entry point of the query, then two different types of primitives are available for navigating in the schema. The first one allows the user to follow paths of concepts, the other one is used for comparing two concepts which are not directly connected to each other. The presence of a main concept associates a type to each query; for example, if the main concept is the entity PERSON, the result of the query will be a set of persons (possibly enriched with attributes of other
partly from the available technology. Graphical systems represent concepts by crude approximations of drawings and limitations, deriving partly from their own paradigms and in order to recognize the reality of interest from its computer a significant part of it) is needed to be able to express the whole language, where usually the learning of the whole language (or
colours, multiple windows, etc., thus extending the expressing a query, where process icons or navigation in an user ase In fact, visual languages use the Full power of new
tec!no!logies, such as two/three dimensional representations, to denote several services and locations in an airport. In information processing systems we distinguish icons denoting objects from icons denoting actions and processes; the first ones are easily understood since they are stylized reproductions of the objects, the second ones are generally more abstract and difficult to understand. In an attempt to formalize a theory of icons, Chang has introduced the concept of generalized icons [3], which are either object icons, representing objects, or process icons, representing actions of computational processes (procedures, etc.). In both cases the icon Xi of the icon (the image as seen on the screen) from the semantic part Xim (the meaning that such image conveys). An example is the well known icon indicating a restaurant, whose pictorial part is the image of a fork and a knife and its semantic part reveals the presence of a restaurant. Since an icon belongs to the bi-dimensional world, where the information is represented by form, structure and colour more than by numbers or text, we will have all the advantages of an icon representation: 1) the expressive power of the figure (as in illustrations, maps, plans, etc), 2) the diminished mental load due to the limited amount of information to be stored (as compared to a long string of alphanumeric characters) and 3) the cultural independence from the viewer since icons have a loose connection both with the user natural language and his specific background.

Icon composition is the basic querying mechanism in iconic systems. For example, in [4] the query is formulated by composing a visual sentence, according to a visual syntax; for example, icons are vertically combined to denote conjunction (logical AND) and horizontally combined to denote disjunction (logical OR). In order to be effective, the proposed set of icons should be understandable by the majority of the public, that the purpose of inviting actions have mainly to meet such a requirement, while icons referring either to object classes or to properties of such classes are chosen with some breadth. However, in many cases it is difficult and often unfaulty to find a universal accepted set of icons. As an alternative, icons could be easily combined to tailor their shape to the particular needs of the user and to his own mental representation of the tasks and methods he wants to perform.

4. Basic Ideas

The considerations of the previous sections motivate the opinion that visual query languages represent evolution, in terms of friendliness, with respect to traditional languages. In fact, visual languages use the full power of new technologies such as two/three dimensional representations, colours, multiple windows, etc., thus extending the man/machine communication bandwidth in several directions. At the same time, the system is represented in a more natural way, shortening the interpretation path the user must perform in order to recognize the reality of interest from its computer oriented representation. The same holds for the activity of expressing a query, where process icons or navigation in diagrams substitute names of commands. Furthermore, visual languages are more flexible than traditional monolithic languages, where usually the learning of the whole language (or a significant part of it) is needed to be able to express the whole range of queries.

On the other hand, visual languages suffer from some limitations, deriving partly from their own paradigms and partly from the available technology. Graphical systems represent concepts by crude approximations of drawings and have a weak metaphorical power so that remarkable efforts may be necessary to obtain a mental reconstruction of the original objects on the screen. Moreover, if the system to be represented has many inter-connections, screen clutter may show up so that a confusing pattern is generated. Furthermore, it may be necessary to express other, more elaborate, concepts so that drawings using only boxes and arrows may be unsatisfactory and new pictorial representations, like pictures and icons, are required. Conversely, no standard can yet be imposed on the set of icons used in different applications. Consequently, as the number of icons increases, in particular when the piece of reality involved in the information system is either complex or specialist, potential ambiguity arises in the interpretation of the set of icons since the discrimination power tends to decrease. In this cases graphical systems, while offering a less immediate representation, do not suffer from such saturation phenomena. Finally, the limits of the available technology do not allow visual languages to exploit all their inherent power. Limitations in both screen size and definition impose to use cluttered drawings or images, thus increasing their reliability; scrolling mechanisms may override the problem, paying the price of loosing a comprehensive representation of the whole reality of interest.

Since the graphical and the iconic approaches seem to present a notable number of complementary aspects, an integrated approach provides, in principle, the best interface for the majority of users, since it combines the advantages of both the original paradigms, and it tries to limit their drawbacks. Moreover, in the process of query expression, the user could commute between the two representations, according to his preferences and his needs. Our proposal is in this direction: in providing, we aim to create a new system QBID*, introduced in Section 2, with an iconic counterpart, obtaining an integrated environment. Both the graphical and the iconic representations of an Entity-Relationship (E-R in the following) schema have to be used in a consistent way. However, the mapping between an E-R schema and the corresponding diagram is immediate, while the iconic mapping is as follows: - every entity is represented by an icon; - every relationship involved in the information system is either graphically or iconically or in a mixed way, alternating the two representations.

It is worth noting that representing an entity by an icon follows immediately from the idea of object icon as an image conveying the meaning of a class of objects. On the other hand, among all the relationships populating an E-R schema, we choose to explicitly represent only the most complex ones, i.e. we hide those specifying only binary links between entities. In principle, the mapping between entities and icons should be stated by the Data Base Administrator using a specific tool. The tool will allow icon editing, icon library management, and mainly both an automatic and a manual mode for associating an icon to either an entity or a relationship. In the automatic mode the tool will be supplied with an ad hoc knowledge base and a learning mechanism, while in the manual one will guide the user only in defining icons which are sound with respect to the schema structure.

Finally, in order to distinguish between the various types of icons (object icons and function icons) different frames surrounding them are used. Figure 1 shows the two frames: the first one represents a class of objects (e.g. an entity), the second one a function or process (i.e. a command).

![Diagram](Diagram.png)

**Figure 1:** The two icon types and corresponding frames

The basic idea is that the same set of query may be expressed either graphically or iconically (or in a mixed way, alternating the two representations). At this aim, the visual operations performed by the user are formally expressed by means of the constructs of a textual language (defined for the graphical
operations into [1]), whose meaning is given in terms of the relational algebra. This choice allows the user to quickly switch between the interfaces, whose similarities are pointed out also in the query structuring (e.g., schema understanding session with the possibility of "cutting" the subschema of interest; idea of main concept as entry point of the query; navigation among concepts).

5. System Architecture

In this section we present a general overview of the proposed query system, including the icon generator tool introduced in Section 4, called ETI (Entity To Icon) in the following. In Figure 2 the major functionalities of the system together with the information flows and the main libraries are shown. Note that, as stated in Section 4, QBD* is placed at the same level of the Iconic Interface (ICI in the following), therefore that the user has two equally expressive ways for posing the query.

![Figure 2: System Architecture](image)

Both ICI and QBD* queries, since are formally expressed by means of the same underlying textual language, are translated first into relational algebra expressions and then in terms of the target DBMS, by the same modules. Moreover, the information on the E-R schema structure are stored into a common library, thus avoiding inconsistencies. For example, if ICI has to retrieve all the icons connected to a specific one via relationships, the working steps are:

- ICI searches for the entity corresponding to the given icon in the E-R Icon Mapping library and for the entities involved in the relationships in the E-R library;
- using the mapping library again, ICI finds the icons corresponding to such entities;
- finally, the corresponding pictorial images are retrieved from the Icon library and shown to the user.

Note that both the icon and mapping libraries are created and updated using the ETI functionalities, illustrated in Figure 3. Such functionalities are expressed by two mappers and an Icon editor, that enables the user to create or modify an icon, so updating the icon library. Such a library constitutes, together with the E-R library, the input for the E-R Icon manual mapper, that allows the user to manually specify the correspondence between entities and icons, providing some help in performing a guided mapping through hierarchies and relationships. Indeed, the user shall associate firstly the entities at the maximum level of generalization with the corresponding icons, then the system will iteratively show the descendant entities in the hierarchies and ask for the related icons. Moreover, during the mapping, the user may view the entities related via relationships to the current one. The E-R Icon automatic mapper is the most complex function of the system, and it is presently in the first stage of definition. The basic idea is to perform a search in the knowledge base in order to automatically associate icons to entities. The E-R library can be updated by both QBD* and ICI during the querying process. In particular, the user may save the query as a new concept, so resulting in the addition of both a new entity to the current E-R schema and a new link to the E-R Icon Mapping.

![Figure 3: A refinement of the ETI functionality](image)

6. More about ICI

In this section we further detail the Iconic Interface. Looking at Figure 4 we see that ICI is composed of three processes and several data stores; two of such processes, the Know And Select (KAS in the following) and the Iconic Query Composition, cooperate. The former yields a set of tools for understanding the information content of the database and, at the same time, allows extracting the icons of interest; the latter processes the query based on the selected icons. The user will rarely be aware of all the icons to be selected, thus during a query session it is possible to switch between the two processes. The third process, Query Icon Mapper, concerns the management of the query stored by the user into the query library. As we said above, once the user has completed a query, such a query may result into a new entity in the E-R schema together with a new related icon. This icon should result from a composition process of existing ones, guided by the Query Icon Mapper.

![Figure 4: A refinement of the ICI functionality](image)
6.1 The Know And Select Process

The Know and Select process allows the user to choose the desired icon set and to know the structure, at the intentional level, of the database. The selection phase is useful to reduce the icon set the user has to deal with, while the knowledge phase is necessary to both discover the links between icons and to deepen the user knowledge about the schema contents. In Figure 5 is shown how the system allows to examine the icons connected to a selected one.

Note that the window is split into several horizontal bands: each one corresponds to the links possibly inherited by the ancestors of the current icon of interest. For example, Figure 5.a shows the links of the icon MAN: in addition to the icons BOXER-SCHOOL, the icon FRUIT and the couple CITY, CITY, inherited by the parent icon PERSON, are displayed. The corresponding E-R schema is presented in Figure 5.b. Note that if more than one link exists between two icons, the system displays, in the corresponding band, as many instances of the same icon as the number of links. Acting on any of the above icons, a window containing more information about the meaning of the corresponding link will appear (see Figure 5.c). Generally, the user can obtain information about properties of the selected icon double clicking on it. Such a command returns a window containing the textual description of both the icon and its properties.

6.2 Iconic Query Composition

As mentioned before, the Iconic Query Composition (IOC in what follows) process enables the user querying the database. The left part of Figure 9 shows the main window of the process, where the command icon space is in the upper layer. The whole window is split into three bands, where the first two generate new icons and are used for the query formulation, while the third one allows for browsing the result. Generally, a query is composed by several subqueries, each one built through standard mechanisms, connected by the usual set operators. The leftmost band commands, characterized by the path-search icon and the set-operator icon, allows for both such activities; in particular, the path-search icon is suitable for building a subquery and the set-operator icon is useful for combining two subqueries. Note that the previous activities have the same syntactical structure in that they operate on two icons, producing a new one. The second band is used for stating conditions on the class of objects represented by the icons (filter command icon). In the following examples, we refer to the E-R schema shown in Figure 7. The associated set of icons is shown in Figure 8.

The basic idea in the query formulation strategy is in enabling the user to focus on two concepts: the main and the target concept of the query. The former is the concept which the objects constituting the final result of the query are instances of, while the latter is any concept related to the main one, whose properties are relevant to the result. The user specifies these two concepts by selecting the two corresponding icons, i.e. the main and the target icon. Typically, several relationship paths in the database may exist between the main and the target concept, and each path may result in a different answer to the query. In order to specify the corresponding correct link between the main and the target icon two different strategies are provided by the IOC, both activable with the path-search command icon. The first strategy, referred to as automatic mode, is adopted when the user wants the system to provide all the existing paths between the main and the target icon. The automatic mode strategy is triggered by moving the target icon to the box placed at the right of the main icon and actin on the path-search icon (see Figure 9, where the main icon is MAN and the target icon is MEDICAL CENTER). At this stage a new window is shown (see Figure 10). The window is split into several horizontal bands, each one concerning an existing simple path between MAN and MEDICAL CENTER.

Figure 5: The Related icon window

Figure 6: The View Property window

Note that the meaning of the schema is strictly joined to our available set of icons. By simple path between two icons we mean any simple path connecting the two corresponding concepts in the E-R diagram.
The E-R schema

Figure 7: The E-R schema

Each band contains all the icons involved in the path, plus a textual label constituted by the names of the E-R concepts involved in the path, chained according to the semantics of the path. This can be useful to further specify the path, so to solve possible ambiguities (e.g. two different relationships between the same entities). The lower band enables the user to link the main icon to the target icon regardless existing paths between them. Such a link is established by comparing the attributes of the two icons by means of the compare-properties icon.

The second strategy, called manual mode, is followed when the user is interested in non simple paths between the main and the target icon. In this case the user explicitly selects all the icons constituting the path between the main and the target icon. In this case, the window Related icons (see fig. 5) drives the user in selecting the icons to be included in the path.

As we said before, all the previous activities result in defining a new class of objects and an associate icon that appears in the box under the main and the target icon. The new icon is built according to a set of iconic composition rules [3]. Finally, in order to further restrict the instances of the new class, conditions may be stated by dragging the new icon in the box under the filter icon. During such an activity it is possible to modify the structure (i.e. the properties) of the class by restricting the set of attributes characterizing the class itself.

Figure 9: The IQC window using the automatic mode

According to the strategy introduced in Section 2, once the user has built two subqueries starting from the same main icon, i.e. two classes of objects of the same type as the main icon has been generated, he can combine them by a set operator. At this aim the user moves the two icons corresponding to the new classes to the leftmost band of the IQC window, acting on the set-operator icon and specifying the operation to be performed (union, intersection, and difference). The resulting class of objects has the same type as the main icon, and its properties are defined as follows:

- in the case of the difference operator, the new class is characterized by the properties of the minuend class;
- in the two other cases, the new class is characterized by the union of both the properties of the starting classes. Note that, in the union case some attributes of the resulting class may be affected by null-values.

Finally, by means of the rightmost band of the IQC window, it is possible to examine the content of an icon, browsing its instances.

203
7. Conclusions

The hybrid approach, i.e. the one combining diagrammatic and iconic interaction modalities, has been shown to be the most suitable to various classes of user and applications [2]. Based on this consideration, we propose a visual query system that allows the user both to express the whole query in either the graphical or iconic modality and to switch between the two modalities during the formulation: the state of the query is consistently updated in both the modalities according to the semantics of the user actions. For example, if the user, working with the iconic interface, first drags an icon into the main icon space, and then switches to QBD* (diagrammatic interface), the relationship HOLIDAY in Figure 7: the iconic system provides a limited view of the E-R formalism.

As an example of a complete query formulated in the two modalities, consider the following: "Find all the men attending a boxer school". Using QBD* the following steps are to be executed:

1) enter the navigation session;
2) select MAN as main entity;
3) select the path: ATTENDS - BOXER SCHOOL;
4) close the navigation session.

On the other hand, using ICI the steps are as follows:

1) active the IOC process;
2) drag the icon MAN in the main icon space;
3) drag the BOXER SCHOOL in the target icon space;
4) active the path search command;
5) choose the right object class from the large set shown by the system.

Acknowledgements

Support from EEC, under the ESPRIT Project 5638 "KIM", from National Research Council (CNR) Italy, under Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 6, Torna INFORIK, and from Arthur Andersen & Co. Management Consulting is gratefully acknowledged.

Moreover, we would like to thank Enrico Melis and Pier Angelo Sottile for many extremely helpful suggestions, ideas and comments.

References