# PRINCIPLES OF AN ICONS-BASED COMMAND LANGUAGE

C Frasson and M Er-radi
Département d'informatique et de
recherche opérationnelle
Université de Montréal
C P 6128, Succ A
Montréal, Québec
Canada H3C 3J7

ABSTRACT

Improvements both in technology and in user-oriented software have shown the feasibility of new kinds of non-procedural languages. However, interaction between end-user and data should rely more and more on graphical languages and, particularly, on 'iconic" languages. In the following we review and analyze the forces which are at the origin of changes in the user environment. We give the main specifications of an iconic interface and a command language based on icons. Examples are given in a medical environment.

Keywords databases, icons, direct manipulation, user interface.

## 1.- INTRODUCTION

When we consider the evolution of programming languages, we generally notice that each improvement in computer technology allowed satisfying new needs and a growing number of users. Using those improvements new software and new methods were developed [FRAS85]. To date four major language generations can be distinguished.

Early computers were intended for specialists (scientific and engineering applications). They used machine languages (first generation) or assembly languages (second generation) which both depended on the type of the computer. Club membership' was limited to a few initiated persons.



The need for more readable and understandable programs and the second generation of hardware led to high-level languages (third generation language) and to new programming methods. Compilers took over translation into machine language through several intermediate steps. Thus, languages such as Fortran, Cobol, PL/1 and later Pascal, Basic, allowed a better productivity and, in certain cases, a structured approach to problem solving

However the programs developed depended on file organization techniques and access methods. Periodic changes resulted in a maintenance activity and thus in a waste of programmer effort. So the need for data independence, introduced by Date [DATE71], led to database management systems [DATE82] with non procedural languages.

Also, improvements both in technology and in user-oriented software have shown the feasibility of new kinds of languages. They use high level operators capable of manipulating entire sets as single objects instead of being restricted to one record at a time. In certain cases they span a wide range of users so that they can satisfy both professional and end-users requirements

Applications development systems like SQL/DS, IDEAL, NATURAL, FOCUS, RAMIS,... attempt to integrate all development services in a user friendly manner and constitute the fourth-generation languages.

However, interaction between end-user and data should rely more and more on graphical languages and, particularly, on iconic' languages, which would constitute the fifth-generation languages. After reviewing the forces which are at the origin of changes in the user environment, we highlight the interest for an iconic interface. We give the main specifications of a command language capable of expressing, through a manipulation of icons, direct queries to a relational database. We give the components

of the system (called Icons-Based System) and we illustrate some of its capabilities.

Before examining the features of our iconic language let us see why user requirements have (recently) evolved.

## 2.- THE CONVERGING FORCES OF THE USER'S ENVIRONMENT

### Databases

Until the third generation the tools developed around the computers were intended for professionals who used a programming interface to express their needs. This interface progressively evolved with database systems and particularly with the relational model [ASTR76], [CHAM80]. The hierarchical [IMSV74] and network [DBTG71] languages were, indeed, too close from the procedural languages. Only the relational model allows defining. and manipulating data without having in mind the logical or physical organization [CHAM80]. Some relational languages like SQL (used in IBM's SQL/DS system and in the oracle system [RSI79]), Dbase III [DBAS84] or knowledge-manager [KNOW83], have two layers. One provides an immediate means for retrieving or manipulating data using very simple and powerful commands which can be issued by end-users. The other layer is intended for programmers with procedural statements also including powerful commands.

Several facilities have been introduced to facilitate the man/data communication:

- explicit report generators.
- screen-management software.
- data dictionaries.
- multiples views of the database.

Their introduction corresponds to a need for every system which can be easily used both by end-users and professional programmers.

### Micro-computers availability

As the hardware cost decreases and micro-computers can now be purchased at a very low price, more and more people today are wishing to use a computer for their needs. However they are frustrated by the necessity to learn a programming language for expressing those needs. Thus, there is an increasing demand for software able to facilitate the communication between end-users and the computer.

### Office automation systems

The office environment is particularly relevant. It involves non specialized users who generally wish a ready-to-use and simple system which does not require technical knowledge [TSIL80], [ELBE82], [LEWL84]. Spectacular results can be obtained when the system satisfy acute needs. (Visicalc is such an example). Many requirements (word processing, electronic mailing systems, electronic cabinet, ...) should be satisfied using adequate tools with high-level interface [ZLOO82].

### Videotext

Communication networks provide facilities for sharing data and accessing them through formated screen. A sophisticated interface is also needed here to allow immediate use by a wide variety of people.

### Need for productivity

Many techniques have been developed in order to improve productivity of program development (structured programming, interactive techniques, program generators, ...).

However improvements due to these techniques have been modest and even insufficient when taking into account the time wasted in maintenance and system design. New computer languages, which improve the productivity of programmers and provide a direct and powerful interaction with the end-user, have recently appeared. The fourth-generation languages involve database languages such as SQL/DS (IBM), Oracle [RSI79], Rbase 4000 [RBAS84], Dbase III [DBAS84], knowledge Manager [KNOW83], ... and system builders like Ramis II [RAMI83], Mapper [MAPP83] which allow the user to define his problem and to find an adequate solution through interaction with the system.

## 3.- THE ICONIC INTERFACE

As we pointed out earlier, an end-user who understands his own needs wishes a direct interaction with the computer. He wants to develop his own system and to query without having to worry about procedural details. Fourth generation languages constitute a first step in that direction.

The advantage of a graphical query language was first noted by Zloof with QBE [ZLOO75] and Stonebraker with CUPID [MCDS75]. Similar ideas were introduced by CHANG [CHAN80], [CHAN81] for managing pictures.

In the "Spatial Data Management System", introduced by Herot [HERO80], [HERO82], the information itself is expressed in graphical form and presented in a spatial framework providing a more obvious structure than in a conventional DBMS.

Other works were motivated by the increasing need for more sophisticated user interfaces [ZLO082], [PUFK83]. A user-interface using interactive techniques, like the mouse and the bit-mapped display [SIKV82], [ELBE82], enables users to perform their tasks directly and easily [SCHN82], [SCHN83].

Recent introduction of new functionalities such as windowing, selecting and manipulating objects with a mouse [WILL83], allows defining and using a symbolic representation (an icon) for various purposes.

An icon can be sufficiently expressive. The international system of road signs is an example of icons that everybody can easily understand. In various domains (medicine, scientific applications, mechanic, traffic control, trade, ...) icons are used for communicating with groups of users. Icons were widely used by the egyptians several centuries ago. The objective is to facilitate the interaction between the users and their environment by appealing to an immediate understanding of the information represented by the icons. Curiously an obvious human capability has so far (rarely) been exploited: the human brain is more easily capable of processing picture than text. It seems that about 80% of human knowledge is based on visual perception. An image is more explicit and representative of a domain than several hundred words describing this domain.

In [GLTA84] the authors note that programmers "often attempt to transform the human mind's multidimensional, visual and often dynamic conception of a problem's solution into the one-dimensional, textual and static representation required by traditional programming languages". They propose a programming language (PICT) which can assist programmers by providing a set of graphics to compose, edit and run programs through a menu of icons.

An interesting analysis of visual languages is presented in [SHU85]. In [RAED85] the author shows how pictures can yield benefits throughout the phases of a programming project.

Our approach is based on an interactive command language which uses an iconic interface particularly intended for end-users, for directly handling both alphanumeric data and images. In the following we describe its main characteristics.

Instead of using a tabular form of query formulation, as in QPE [CHAN80], we use icons to sketch an example which will serve to retrieve or update relational data.

## 4.- PRINCIPLE OF AN ICONIC COMMAND LANGUAGE

The Icons-Based System (IBS) which we describe here consists of two main components:

- an object interface;
- an icons-based set of commands

### The object interface

This interface is made up of a set of objects familiar to the user and of a set of operations which can be performed on these objects. Objects and operations are arranged on a screen like on a desktop. We distinguish three areas (figure 1):

- the workspace area where the objects of a given database can be manipulated through a window
- the information area contains the different properties that an object can have. These properties can be scrolled up or down
- the functional area contains the different types of operation which can be performed on the objects;

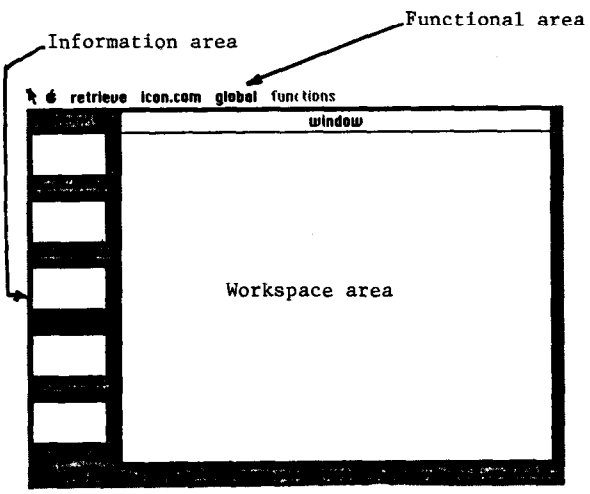
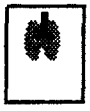



Figure 1 The area of interfaces

The selection of an object or an operation is done by direct manipulation, for instance with a mouse as in several other systems [PUFK83], [WILL83]: pointing on the object or on the operation (in a pop-up menu) and clicking on the mouse. In the following we will refer to those two combined actions as "select".

Different types of objects may be manipulated. They concern alphanumeric, voice, graphic and image data.

An icon is a graphical representation of an object. In [FRER85] icons are both associated with three types of objects in a medical environment: alphanumeric data, images, and voice. For instance the following icons illustrate information concerning patients and lungs.



It is important to notice that even for a non specialized user (for instance a user who is not familiar with a medical environment) the meaning of an icon becomes evident almost at first glance. An object can itself be composed of others objects. For instance a heart, a jaw, are composed of several subsets.

In addition it is possible to associate properties with each object and we can also represent properties by icons when they have a meaning from a spatial point of view. The structure of the database is displayed in the workspace area by selecting an operation like "SHOW STRUCTURE" in a pop-up menu of the functional area. Then, selecting a particular object of this database will send it into the workspace area with its properties:

- if the object is alphanumeric (for example a patient) all the corresponding properties are displayed in the workspace,
- if the object represents an image (a lung) the set of properties fills in the information area. In fact only a subset appears in this area but the set can be scrolled,
- if the object contains both alphanumeric and graphic properties they are displayed in the respective areas.

In the example of figure 2 the properties which can be associated with lungs are some typical diseases which can affect a lung: cancer (represented by the icon "."), broncho-pneumonia (  ), perforation (  ), oedema (  ).

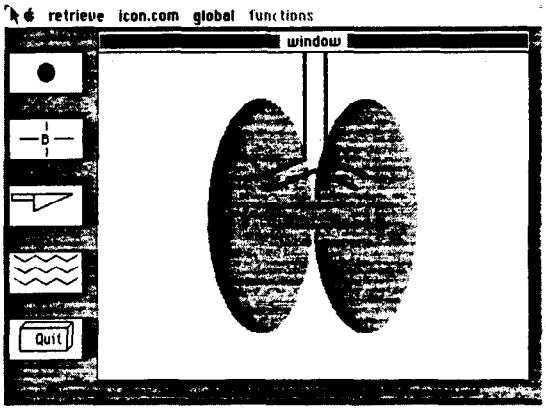



Figure 2 An object with its properties

So, the properties can be attached to a lung by selecting the corresponding icon in the information area, then by clicking on the desired position (on the lung) in the workspace. The final drawing can be manipulated by selecting a specific command in the functional area. It is also possible to include textual data like comments (diagnosis) somewhere in the workspace.

The icons-based manipulations

The command language is essentially based on icons and commands manipulations. Some manipulations can be achieved, as mentioned above, with an interactive device. In addition a set of operations can be applied on the object present in the workspace. These operations are displayed in the functional area, in pop-up menus. We distinguish information processing commands and functions processing commands. For the moment, the following commands are considered in IBS:

- INFORMATION PROCESSING COMMANDS: they include Icons, Retrieval and File commands.

Icons commands

CREATE : creates an icon for a new object. The process can use drawing facilities (like in Mac-Paint) or a set of basic figures which can be composed.

MODIFY : modifies an icon.

DELETE : erases an icon from the iconic interface. (Thus it is possible to obtain a subset more suited for an application).

ZOOM : enlarges a part of the object according to a given scale.

REDUCE : reduces the size of the current object.

ROTATE : performs a rotation of the current object.

Retrieval commands

QUALIFY : allows saving properties assigned to an object present in the workspace (by clicking as explained above).

CUT : eliminates a property from an object.

SEARCH : finds an object satisfying a given set of criteria (alphanumeric or graphic criteria).

SEARCH LIKE : uses an existing object as a target to find similar instances.

HIGHLIGHT : highlights icons satisfying certain conditions in a composed object.

HIDE : hides elements of the object present in the workspace and satisfying certain conditions.

INTEGRATE : adds a graphic criterion to a current object. The resulting object is then composed.

UNDO : undoes the consequence of the last operation.

LINK : allows binding a set of selected instances of an object to another set, according to a specified relationship (or the existing relationship if not specified).

File commands

SHOW STRUCTURE: gives the iconic representation of the objects in the database and their relationships.

SHOW TYPES : displays all the objects available in the system.

SHOW CURRENT : shows the last object type retrieved.

SORT : sorts the different instances of an object according to alphanumeric criteria.

COMMENT : includes a textual comment in the object. Also it is possible to modify the current comment using direct access to the text or a word processing system.

IMAGE : gives the actual image of the object represented by an icon and recently selected.

Function processing commands

EXTRACT : selects an operation from the set of all the operations available and includes it in the system of the user (functional view).

SHOW FUNCTION : shows all the operations.

CREATE FUNCTION: includes a new operation (programmed function) into the system.

RANGE : allows specifying a range of spatial retrieval (coordinates) by direct pointing.

All the operations mentionned above are performed using mouse facilities. Non-authorized manipulations are directly controlled by the system itself.

## 5.- SYSTEM CHARACTERISTICS

The main components of IBS are shown on figure 3. We distinguish an iconic interface, a query translator and an iconic database.

- The iconic interface is the subset (a view) of all icons and commands available for the user in an application. They are extracted from the set of basic commands available in IBS and mentionned above. Also, icons used for an application can be directly extracted from a set of basic icons or modified using an icon editor.

- The iconic database contains all the icon (views) used by the system for the applications, and two tables, IAT and ICT, defined as follows:

  IAT (Icon-id, Sub-icon-id)
  ICT (Icon-id, Icon-name)

  where Icon-id and Sub-icon-id are symbolic addresses of icons in secondary memory. They play the part of arrogates which are system generated identifiers [GROS84]. Icon-name is the name of an icon (a lung or a cancer, for instance). The first table allows to associate different properties (Sub-icon-id) to an entity (Icon-id). The second table associates the address of an icon with its textual meaning (for instance: 11, lung). These two tables are used for assigning properties to an object in the workspace area and for translating the resulting sketch into a relational tuple expression.

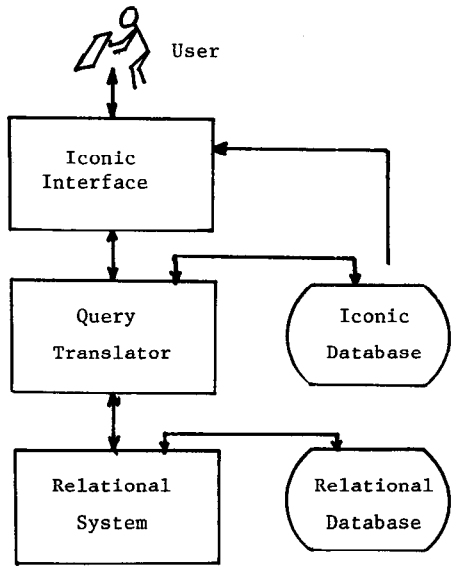



Figure 3 The IBS Components

- The Query Translator translates a direct query (expressed by the user by icons manipulation) into a relational query. The iconic database is used to substitute iconic data with textual data. Positions on the icons are automatically indicated by the pointing device and given to the translator together with alphanumeric data and commands.

  The result of a translation consists in a QBE-like-table where attributes are automatically filled with the previous values.

Finally, the relational query is sent to the relational DBMS and the database. In the answer, relational tuples are converted into iconic or textual representation and given to the user through the iconic interface.

## 6.- EXAMPLES

To illustrate some capabilities of our system we will use a set of objects concerning organs (heart, lungs, stomach, jaws, ...) of a patient. In a classical relational database we might have the following relations:

PATIENT (Pat-no, name, address, birth-date).

STOMACH-DIS (Pat-no, dis-no, disease, diagnosis, date, X1, Y1, width)

LUNG-DIS (Pat-no, dis-no, disease, diagnosis, date, X1, Y1, width)

PICTURE (Pat-no, dis-no, frame)

In the STOMACH-DIS and LUNG-DIS relations, we have the identification of a patient, of a disease, the disease description, the diagnosis issued by the physician, the date of the patient's visit, the coordinates of the disease and the width of a region where the disease occurs.

The PICTURE relation gives the frame number of the patient's actual image containing a disease.

Clicking on the SHOW-STRUCT function (in the "global" menu) displays the different kinds of objects available (figure 4).

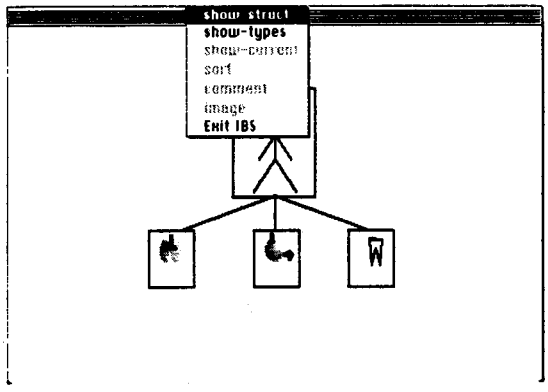



Figure 4

If we select the "stomach" and we decide to assign an ulcer, we select the corresponding icon then we drag it on organ. We obtain the result indicated on figure 5. Integrity constraints can easily be implemented to avoid an incorrect position of the disease. The coordinates of the disease are automatically calculated by the mouse. The diagnosis may also be introduced.

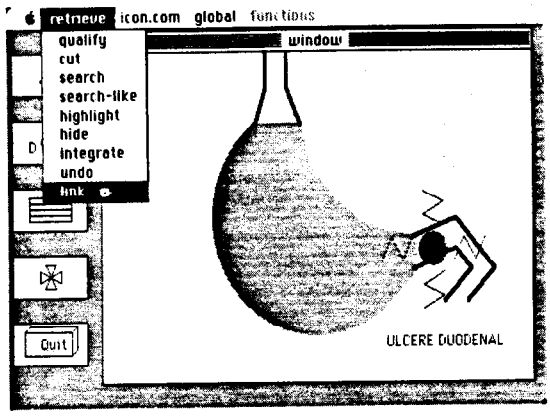



Figure 5

The picture can be linked to a patient by using the LINK Functions and by selecting the patient icon. The name of the patient will be filled in by the user. Depending on the retrieval command selected, the succession of these manipulations will be translated into corresponding relational expressions.

For example let us consider the following query. "Give all the lungs of the patients having the same stomach disease as Eric".

In SQL the request can be resolved as:

```
SELECT DISEASE
FROM LUNG-DIS
WHERE PAT-NO IN
   SELECT DISEASE
   FROM STOMACH-DIS
   WHERE PAT-NO IN
     SELECT PAT-NO
     FROM PATIENT
     WHERE NAME = 'Eric'.
```

In IBS we just select the patient icon which brings the names of the different attributes into the workspace (figure 6). The value 'Eric' is given to the attribute "name" and the SEARCH command is activated (it is possible to indicate several criteria using relational operators).

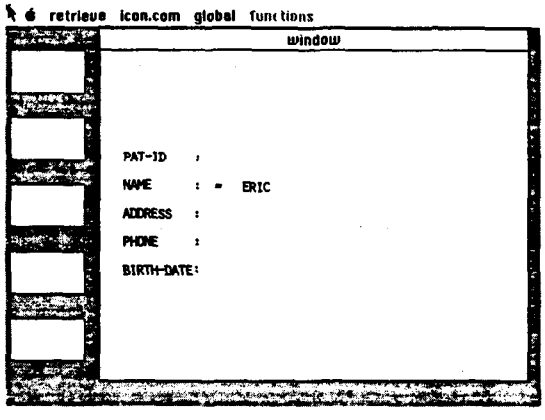



Figure 6

Then, we select the LINK command to bind the result to the following operations. Clicking on the stomach icon will select Eric'Stomach (figure 7).

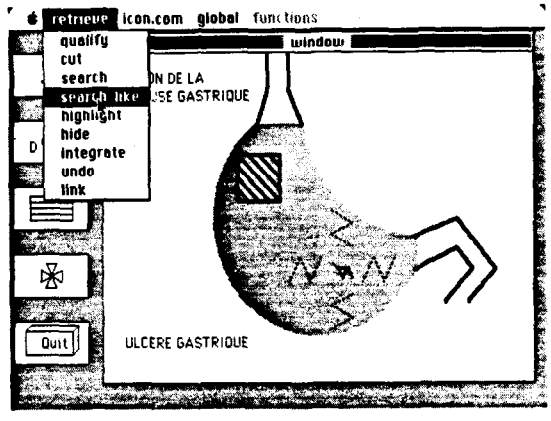



Figure 7

In order to find identical occurences we select the SEARCH LIKE command. Then, selecting the LINK command and the lungs icon will give the result (figure 8).

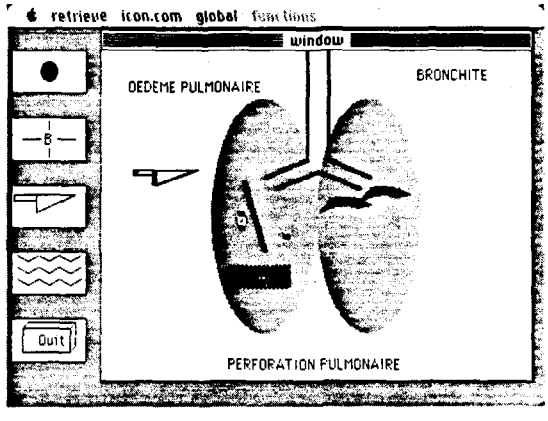



Figure 8

SEARCH LIKE and SEARCH are different commands. With the SEARCH command it is possible to specify the range of retrieval (in the "function" menu). So, the user can specify a value (which will correspond to the width attribute in the relations mentionned above) or he can indicate a region by directly encircling it.

If the user needs the actual image of an organ (for instance the organ retrieved in figure 8) he simply has to select the Image command in the global menu (figure 9).

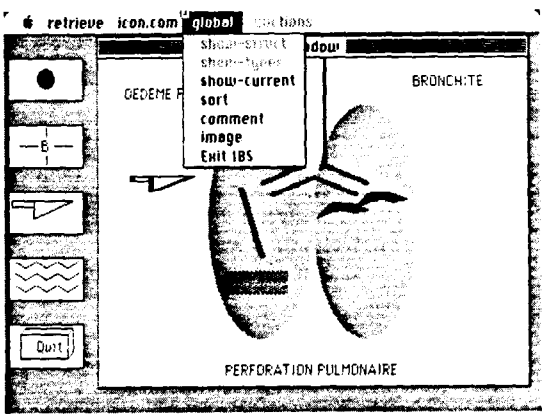



Figure 9

An important characteristic of IBS is that the iconic occurences of a patient (lungs, stomach diseases) do not exist. They are reconstructed from the relational database and visually presented to the user.

Some aspects of the command language have been implemented on a Macintosh by two teams of students. One team used Mac-Forth and the other one used the C language using file access techniques. Results obtained with the language were more efficient. Presently, the system developed in C is currently being integrated with a relational system, via an interface, on a Sun.

## 7.- CONCLUSION

The system and the command language we have presented here are intended for end-user. They use an object-oriented interface and a direct manipulation. Preliminary tests have shown an easy and rapid manipulation. The sequence of different icons or functions selections are done very quickly. We are now examining its extension to other domains of applications.

## - ACKNOWLEDGMENTS

We would like to thank the numerous students who have participated with enthusiasm to the preliminary stages of implementation: S. Decarie, C. Blouin, D. Funke, S. Michel, P. Marchand. We also thank P. McKenzie for his comments. This research was supported by the NSERC under grant A0196.

## REFERENCES

[ASTR76] Astrahan M & Ap. "System R: A relational approach to database management" ACM TODS 1,2 Juin 1976.

[CHAN80] Chang, N.S., Fu, K.S., "Query by Pictorial Example", IEEE Trans. Soft. Eng. SE6 (1980).

[CHAN81] Chang. N.S., Fu, K.S., "Picture Query Languages for Pictorial Database Systems", Computer 14, 23-33 (1981).

[CHAM80] Chamberlin, D., "A summary of user experience with SQL data sublanguage", Proc. International Conference on Data Bases, Aberdeen, Scotland, Juillet 1980.

[DATE71] Date, C.J., Hopewell, P., "File definition and logical data independence", Proc. ACM SIGFIDET Workshop on data description access and control, 1971.

[DATE82] Date, C.J., "An introduction to database systems", Addison Wesley, ed. (Third edition) 1982.

[DBAS84] Dbase III [TM] "Dbase III Reference Guide", Ashton Tate, 1984.

[DBTG71] Data Base Task Group of Codasyl P. - Programming language Committee Report (April 1971) - Available from BCS,ACM.

[ELBE82] Ellis C., Bernal M. "Officetalk-d-an experimental office information system - Proc. ACM, SIGOA conference, 1982.

[ERFR84] Er-radi, M., Frasson, C., "Image Database Systems", Dép. I.R.O., Université de Montréal, Internal report 155, October 1984.

[FRAS85] Frasson, C., "The evolution of end-user languages". International Symposium on New directions in Computing, IEEE, Trondheim, Aug. 1985.

[FRER85] Frasson, C., Er-radi, Mohammed., "An Icon-Based Language for Applications in Medicine", Département d'I.R.O., University of Montreal, Internal report 158, January 1985.

[GLTA84] Glinert P., Tanimoto L., "Pict: an interactive graphical programming environment. Computer IEEE, November 1984.

[GROS84] Grosky W. "Toward a data model for integrated pictorial databases", Computer Vision Graphics and Image Processing 25, Academic Press 1984.

[HERO80] Herot C. "Spatial management of data. ACM TODS, Vol. 5, no 4, Dec. 1980.

[HERO82] Herot C. "Graphical user interfaces" Proc. of the NYU symposium on user interfaces, May 1982.

[KNOW83] Knowledge Man, "Reference guide" Micro Data Base Systems, Lafayette, 1983.

[LEWL84] Lee A., Woo C., Lochovsky F. "Office-aid: an integrated document management system". Second ACM-SIGOA conference on office information system, Toronto, Juin 1984.

[LIEN77] Lien, Y., Utter, D., "Design of an image database". Proc. 1977, IEEE, Workshop on Picture data description and management. 1977.

[MAPP83] Mapper 5 et 6 "Reference guide". Sperry Inc., Mississauga, Ontario, 1983.

[MCDS75] McDonald, N., Stonebraker, M., "CUPID, The Friendly Query Language". Proc. ACM Pacific Conference, San Francisco, (April 1975).

[PUFK83] Purvy R., Farrel J, Klose P. "The design of star's record processing: data processing for the non computer professional. ACM Transaction on office information systems, vol 1, no 1, 1983.

[RAED85] Raeder, G., "Programming in Pictures: implications on software development". International Symposium on New Directions in Computing, IEEE, Trondheim, Aug. 1985.

[RAMI83] Ramis II, "Reference guide". - Mathematica products group Inc., Nepean, Ontario, 1983.

[RBAS84] Rbase 4000, 6000 "Reference guide". Microrim Inc., Bellevue.

[RSI79] Relational Software Incorporation "Oracle Introduction" version 1.3, 1979. Available from RSI, 3000 Sand Hill Road, Menlo Park, California 94025.

[SCHN82] Schneidermann B., "The future of interactive Systems and the emergence of direct manipulation". Proc. of the NYU symposium on user interfaces, May 1982.

[SCHN83] Schneidermann B., "Direct manipulation: a step beyond programming language", IEEE Computer. Vol. 16,8, August 1983.

[SHU85] Shu, N.C., "Visual Programming languages: a dimensional analysis", International Symposium on New Directions in Computing, IEEE, Trondheim, Aug. 1985.

[SIKV82] Smith D., Irby C, Kimball R., Verplank B., Harshem E. "Designing the star user interface - Byte vol 7, no. 4. 1982.

[TSIL80] Tsichritzis D., Lochovsky F., "Office information systems: challenge for the 80's - Technical report CSRG-127, Computer Systems Research Group, University of Toronto, March 1981.

[WILL83] Williams. G., "The Lisa computer system", Byte vol. 8,2, 1983.

[ZLOO75] Zloof, M.M., "Query by Example". Proc. National Computer Conference, Vol. 44. AFIPS Press 1975.

[ZLOO82] Zloof. M., "Office by example: a business language that unifies data and word processing and electronic mail, IBM System Journal, vol. 21,3, 1982.