On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

Usability of Visual Information Retrieval Metaphors for Object-Oriented Databases

Mariusz Trzaska (mtrzaska@pjwstk.edu.pl)[*], Kazimierz Subieta[#*]

[#]Institute of Computer Science, Ordona 21, Warsaw, Poland
[*]Polish-Japanese Institute of IT, Koszykowa 86, Warsaw, Poland

Abstract. We present visual metaphors of Mavigator, a visual information retrieval system for Web-oriented database applications. The metaphors allow the naive users (computer non-professionals) to retrieve information from object-oriented databases in an easy and intuitive form. Novel features of Mavigator include coherent combination of a few paradigms: intentional navigation, extensional navigation, baskets for recording retrieval results and active extensions. The latter are programmed in a programming language thus allow a programmer to extend functionalities available to end users with no limitations. Due to the flexible architecture Mavigator is able to work not only with a proprietary object-oriented database but also with any information source. The objective of the corresponding PhD thesis is investigation into usability of different visual retrieval metaphors in realistic Web applications. In this paper we present some of the metaphors and discuss their potential for users.

1 Introduction

Databases, among other features, should provide an easy access to information. However, the "easiness" is relative to a user kind and his/her experience in computer technologies. Nowadays more and more non-professionals use computers, especially various applications based on Web browser-oriented interfaces. Their requirements with respect to user-friendliness of the entire computer systems are much stronger than requirements of computer professionals. Non-professionals usually do not accept formalized syntax, strict sophisticated rules of user action, long and specialized training or following professional manuals. As a consequence, computer tools, including interfaces for information retrieval, must evolve into non-sophisticated and easy-to-use applications.

On the other hand, the interface should be simple, but not simpler than the inherent complexity of the task that the user has to solve. For some tasks the typical full-text retrieval (as e.g. in Google) is not sufficiently precise because of lack of facilities to specify semantic meaning of data items stored on Web. The XML technologies (including RDF, OWL, Semantic Web and other proposals) aim at putting data into nested labelled structures with well-defined semantic meaning. There are also query

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

languages such as XQuery to express the user needs much more precisely than it is possible in full-text retrieval engines.

In the database domain there are many query languages proposed, in particular, SQL as a major language for accessing relational databases. SQL, like all textual query languages, is very powerful, but its audience, due to its complexity, is rather limited to computer professionals. The same concerns more recent textual languages for object-oriented databases, such as OQL [1] or SBQL [2].

Attempts to define more user-friendly interfaces have been noticed for many years. One of the most successful was QBE [3] based on a tabular view of relational tables and specific retrieval conditions inserted by the user into the tables. QBE was perhaps the first graphical query interface based on visualization of a database schema and specific visual manipulations of the user on the graphical interface. Due to a better hardware and popularity of easy application programming interfaces for graphical manipulations we observe recently extensive development of visual metaphors for information retrieval. Some of them are counterparts to their textual predecessors. Other, like Value Bars [4] and the interface presented in [5] are based on specific graphical ideas.

As noted in [6], the key issue behind such proposals is *usability* in real applications prepared for users who are not computer professionals. Unfortunately, usability cannot be predicted in advance during design and implementation of an interface, because it depends on many factors such as the readiness of the user to get some training, inherent complexity of the task that he/she has to accomplish and adequacy of the interface to this task, supporting various forms of user awareness during long sessions, and others. Therefore the only way to check the usability is to implement a particular metaphor and then, to measure some user-oriented factors (such as the number of errors, the entire time to reach the goal, etc.) in real applications.

In this paper we present the visual querying interface Mavigator which has been developed to allow non-computer professionals to work with object-oriented database systems. Its key concepts include: intentional navigation, extensional navigation, persistent baskets for recording temporary and final results of querying, and active extensions. The first kind of navigation (intentional) is based on navigation in a database schema graph. The user moves from vertex to vertex via edges, where vertices denote classes (more precisely, collections of objects) and edges denote associations among classes (in UML terms). Additional functionalities allow the user to build quite complex queries. The second kind of navigation (extensional) is similar but the user navigates in a graph of objects rather than in a graph of classes. This mode enables the user to move (in mind) from an object to an object (vertices of the graph) via a specific link (an edge of the graph).

After the user has retrieved some result he/she may require to present it visually in some friendly way, e.g. as a tabular report, as a chart, as a distribution map, etc. This is a critical issue for visual querying interfaces, as the trade-off between simplicity of the end user interface and complexity of a possible visualization form. The number of options and the general complexity of the interface that may be required to visualize a querying result seem to be unacceptable for naive users. Therefore for achieving the required goal we assume some contribution of computer professionals. The last

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

feature, called active extensions, allows a computer professional to add functionalities required by a particular naive user. We assume that the functionalities will be coded in a programming language (currently C#) thus the range of the functionalities is unlimited.

The remainder of this paper is organized as follows. In Section 2 we discuss related work on visual information retrieval facilities and explain how Mavigator differs from the existing solutions. In Section 3 we give a detailed description of the Mavigator metaphors. Section 4 describes two prototypes: SKGN [7, 8] implemented for the European project ICONS, and Mavigator extending and improving SKGN. Mavigator is currently under construction. Section 5 concerns implementation and architecture of prototypes. Section 6 briefly discusses proposed evaluation procedures and Section 7 concludes.

2 Related Work

Due to the limited space this paper gives only an outlined overview of the related work. The final version of the PhD thesis will contain more detailed description.

Roughly speaking, visual metaphors to information retrieval can be subdivided into two groups: based on graphical query languages and graphical browsing interfaces. This subdivision is not fully precise because many systems have features from both groups. An example is Pesto [9] having possibilities to browse through objects from a database. Otherwise to Mavigator, the browsing is performed from one object to a next one object. For instance, the user can display a Student class object, but to see another student, he/she needs to click next (or previous) button and replace current visualization. Besides browsing, Pesto supports quite powerful query capabilities. It utilizes the query-in-place feature, which enables the user to access nested objects, e.g. courses of particular students, but still in the one-by-one mode. Another advantage concerns complex queries with the use of existential and universal quantification. Such complex features may however compromise usability for some kinds of users and kinds of retrieval tasks.

In our opinion an essential issue behind such interfaces is how the user uses and accumulates information during querying. For instance, the user may see all the attributes even those, which are not required for the current task. Otherwise, the user can hide non-interesting attributes, but this requires from him/her some extra action. Therefore, from the user point of view, there is some trade-off between actions that have to prepare the information necessary for querying and actions of further querying. To accomplish complex queries without putting them explicitly, the system should support any sequences of both types of actions.

Typical visual querying system are Kaleidoscape [10], based on its language Kaleidoquery [11], and VOODOO [12]. Both are declared to be visual counterparts of ODMG OQL [1] thus graphical queries are first translated to their textual counterpart and then processed by an already implemented query engine. The first one uses an interesting approach to deal with AND/OR predicates (another proposal can be

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

found in [13]). It is based on a flow model described previously by Schneiderman [14]. We find it very useful and intuitive thus we have adopted it to our metaphor.

Polaris [15], designed for relational databases, has some querying capabilities, but it seems that the major emphasis has been put on data visualization.

A typical example of a browsing system is GOOVI [16] developed by Cassel and Risch. Unfortunately, selecting of objects is done via a textual query editor. A strong point of the system is the ability to work with heterogeneous data sources.

Another interesting browser is [17], which is dedicated to Criminal Intelligence Analysis. It is based on an object graph and provides facilities to make various analyses. Some of them are: retrieving all objects connected directly/indirectly to specified objects (i. e. all people, who are connected to a suspected man), finding similar objects, etc. Querying capabilities include filtering based on attributes and filter patterns. The latter allow filtering links in a valid path by their name, associated type, direction or a combination of these methods. The manner of work is similar to the metaphor that we have called extensional navigation.

Browsing systems relay on manual navigation from one object to a next one. During browsing the user can read the content of selected objects. Browsing should be an obligatory option in situations when the user cannot define formally and precisely the criteria concerning the search goal.

3 Mavigator Metaphors

Mavigator is made up of four metaphors: intentional navigation, extensional navigation, persistent baskets, and active extensions. The subdivision of graphical querying to “intentional” and “extensional” can be found in [18] and [19]. We have adopted these terms for the paradigm based on navigation in a graph. The user can combine these metaphors in an arbitrary way to accomplish a specific task.

Intentional and extensional navigation are based on navigation in a graph according to semantic associations among objects. Because a schema graph (usually dozens of nodes) is much smaller than a corresponding object graph (possibly millions of nodes), we anticipate that intentional navigation will be used as a basic retrieval method, while extensional navigation will be auxiliary and used primarily to refine the results. Next subsections contain description of the methods.

3.1 Intentional navigation

Intentional navigation utilizes a database schema graph. Figure 1 shows a part of such a graph for the Structural Fund Knowledge Portal implemented within the European project ICONS. A graph consists of the following primitives:

- Vertices, which represent classes or collections of objects. With each of them we associate two numbers: the number of objects that are marked by the user (see further) and the number of all objects in the class,
- Edges, which represent semantic associations among objects (in UML terms),

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

- Labels with names of association roles. They are understood as pointers from objects to objects (like in the ODMG standard, C++ binding).

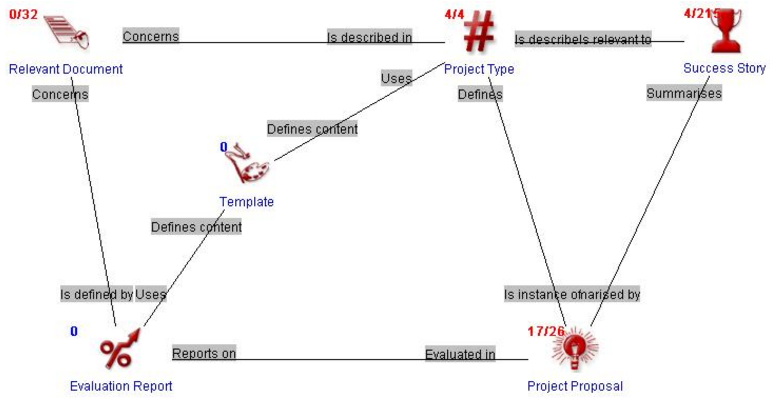

Figure 1. Intentional navigation graph

The user can navigate through vertices via edges. Objects which are relevant for the user (candidates to be within the search result) can be marked, i.e. added to the group of marked objects. There are a number of actions, which cause objects to be marked:

- Filtering through a predicate based on objects' attributes. The action cause marking those objects for which the corresponding predicate is true. There are two options: objects to mark are taken from a set of already marked objects or from entire extension of the class. Filtering objects through a predicate is analogous to the SQL ***select*** clause.
- Manual selection. Using values of special attributes from objects (identifying objects by comprehensive phrases) it is possible to mark particular objects manually. It is especially useful when the number of objects is not too large and there are no common properties among them.
- Navigation (Figure 2) from marked objects of one class, through a selected association role, to objects of another class. An object from a target class became marked if there is an association link to the object from a marked object in the source class. Figure 2 explains the idea. Let's assume that the ***Firm*** set of marked objects has four marked objects: *F1*,…,*F4*. Than, navigating from ***Firm*** via ***employs*** cause marking eight objects: *E1,…,E8.* This activity is similar to using path expressions in query languages. A new set of marked objects (the result of navigation) replaces existing one. It is also possible to perform a union or intersection of new marked objects with the old ones. Notice that the later options allow one to perform a query like: get all employees working in the "Main" department **and** earning more than 5000. The options are easy to understand (even for naive users) and greatly enhance the retrieval capabilities.

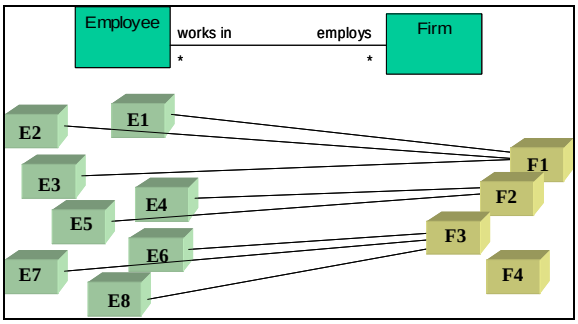

Figure 2. Intentional navigation

- Basket activities. Dragging and dropping the content of a basket on a class icon causes some operation on the marked objects of the class. A new set of marked objects taken from the basket can replace the existing set of marked objects, can be summed with it, can be intersected with it, or can be subtracted from it (equivalents to OR, AND and NOT).
- Active extensions. In principle, this capability is introduced to process marked objects rather than to mark objects. However, because all the information on marked objects is accessible from a C# program the capability can also be used to mark objects.

Intentional navigation and its features allow the user to receive (in many steps but in a simple way) the same effects as through complex, nested queries. Integrating these methods with an extensional navigation, manual selection and other options supports the user even with the power not available in typical query languages.

An open issue concerns functionalities that are available in typical query languages, such as queries involving joins and aggregate functions. There is no technical problem to introduce them to Mavigator (except some extra implementation effort) but we want to avoid situation when excess options will cause our interface to be too complex for the users. We hope that during evaluation we will find answers on such questions.

3.2 Extensional navigation

Extensional navigation takes place inside extensions of classes. Graph's vertices represent objects, and graph's edges represent links. When the user double clicks on a vertex, an appropriate neighbourhood (objects and links) is downloaded from the database, which means "growing" of the graph.

Extensional navigation is useful when there are no common rules (or they are hard to define) among required objects. In such a situation the user can start navigation from any related object, and then follow the links. It is possible to use basket for storing temporary objects or to use them as starting points for the navigation.

3.3 Baskets

Baskets are persistent storages of search results. They store two kinds of entities: unique object identifiers (OIDs) and sub-baskets (Figure 3). The hierarchy of baskets is especially useful for information categorization and keeping order. Each basket has its name that is typed in by the user during its creation. The user is also not aware OIDs, because special objects labels are used. During both kinds of navigation it is possible to drag an object (or a set of marked objects) and to drop them onto a basket. The main basket (holding all the OIDs and sub-baskets) is assigned to a particular user. At the end of a user session all baskets are stored in the database.

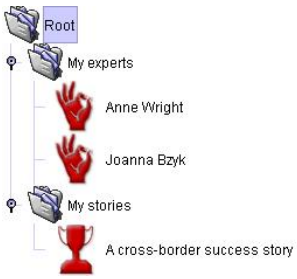

Figure 3. Visualization of the basket containing two sub-baskets and three objects

Below we list all basket activities:

- Create a new basket.
- Change basket properties (name, description).
- Remove selected items (sub-baskets or objects).
- Perform operations on two baskets: sum of baskets, intersection of baskets, and set-theoretic difference of baskets. The operation result can be stored in one of the participating baskets or in a new one.
- Drag an object and drop it onto an extensional navigation frame. As the result, the neighbourhood (other objects and links) of a dropped object will be downloaded from the repository.
- Drag a basket and drop it onto class's visualization in the intentional navigation window. As the result a new set of marked object can be created (replace, add, intersect, subtract). Only objects of that class are considered.

Baskets allow storing selected objects in a very intuitive and structured way. Navigation could be stopped at any time and temporary results (currently selected/marked objects) can be named, stored and accessed at any time.

3.4 Active Extensions

Active extensions provide a way to add new functionalities operating on marked objects or objects recorded in baskets. In order to achieve the maximal power and flexibility active extensions are based on a fully-fledged programming language.

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

Such a solution requires collaboration of an end user with a programmer who will write the code accomplishing the required functionalities.

The prototype, currently under construction, will be based on Microsoft C# as a language for active extensions. A programmer will be aware of the Mavigator metadata environment, which will allow him/her to write a source code of the required functionality in C#. Then the end user will be supported with one click button causing execution of the written code. The code will process the visually selected set of marked objects or objects recorded in a user basket. The functionality of such programs is unlimited. Next two sub-sections present its particular applications.

3.4.1 Active Projections

Active projections allow visualizing a set of marked objects where position (in terms of x, y coordinates) of each of them will be based on value of particular objects' attributes. By using two or three axes it is possible to visualize dependencies of two or three attributes. The number of attributes could be increased due to other visual features. In particular [15, 20, 21] propose to use the features of visual icons such as shape, size, orientation, colour, texture and correlate them with values of additional attributes. For instance, the size of an employee icon could be proportional to his/her salary. Such an approach makes it possible to identify some groups of marked objects. For instance it will be easy to see the group of employees who are paid less than $2000 and have some specified experiences.

Besides the visual analysis of objects dependencies it is also possible to utilize projections in more active fashion. Object taken from a basket could be dropped on projection's surface, which cause right (based on attributes values) placement. It is also possible to perform reverse action: drag object from surface onto the basket (which cause recording object in a basket). We also consider some kind of extensional navigation on projection's surface: clicking on object's visualization causes visualizing its neighbourhood. To avoid misunderstanding, objects from the neighbourhood should be visually distinguished (e.g. because they could have types different from the type of visualized objects).

3.4.2 Objects Exporters

Objects exporters allow cooperating with other software systems. By having a selected set of marked objects, it is possible to send it to other programs, such as Excel, Crystal Reports, etc. That approach makes it possible a subsequent processing of Mavigator's results of querying/browsing.

4 User Interface for Mavigator

There are two implemented prototypes, which follow the Mavigator metaphors. First one, called Structural Knowledge Graph Navigator (SKGN), have been developed during the European project ICONS (Figure 4). It utilizes almost all (except Active projections) concepts described in the previous sections.

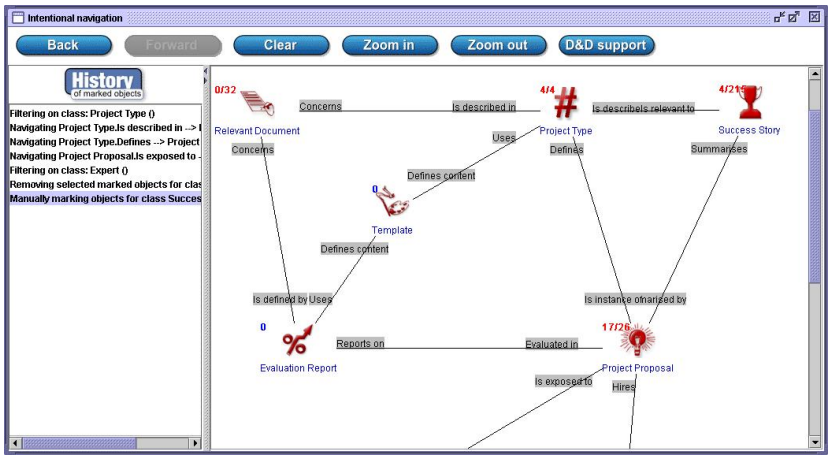

Figure 4. SKGN window during intentional navigation

SKGN has a lot of features, which make them easy to use by casual users. For instance, during intentional navigation all activities involving marked objects could be undone or redone. This is similar to browser back/forward buttons. Each step is stored and shown in the list (see left side of Figure 4). There is also the possibility to go back to any previous step from the recorded history of navigation.

An extensional navigation window allows to utilize facilities such as zoom in/out and scroll bars. All of these functionalities improve navigation and legibility of the graph. Because of legibility, permanent showing of link labels is avoided (however it is possible to turn it on in the user's options). Instead, ontology, auto ontology and specialized tool tips features are introduced. The ontology functionality means that when an object is selected, its class is selected too (on a classes graph – the intentional navigation window). Auto ontology is similar to ontology with one difference – selecting an object's class proceeds when object's tool tip is shown (without selecting the object). A similar functionality has been implemented for links.

In Mavigator we plan to implement all functionalities described above and many others. Some of them are listed below:

- Create a new user interface based on filter flow model [14] for visual definition of predicates.
- Additional activities for marked objects, which could be implemented using Active Extensions (see chapter 3.4). New functions, such as average, maximum, minimum, sum, etc., allow one to perform other kinds of queries.
- Stickers. A user may want to annotate particular object(s). To do this we would like to introduce little stickers, which can hold any kind of information, which are important to users. Stickers will be persistent and assigned to a particular user.
- Extending the utilized data model by generalization. We realize, however, that excessive complexity of a data model could have impact on the easy-of-use and in consequence, on usability.

- Visual joins. They should allow to group objects from two or more classes, for instance, an employee object with a company object. A group of objects would be treated as an ordinary object, with possibilities of filtering, navigating, marking, storing in a basket, etc.

Two later ideas will be carefully evaluated whether they could be acceptable and useful for a sufficiently wide range of users.

5 Software Architecture and Implementation

The Structural Knowledge Graph Navigator has been implemented as a Java applet. Due to the flexible architecture, it is possible to connect it to any data source (through a proprietary wrapper).

The Mavigator prototype, currently under construction, is implemented as a Windows Form Application in C# language. Its architecture (Figure 5) consists of the following elements:

- GUI – contains implementation of the user interface,
- Business logic – includes implementation of the Mavigator metaphors and some additional routines,
- Database wrapper – ensures communication, via defined AbstractDatabase2 interface, with any database. Currently we are working with Matisse [22] post-relational database. In future we plan to develop wrappers to other object-oriented database management systems, including our own prototype ODRA (Object Database for Rapid Application development).

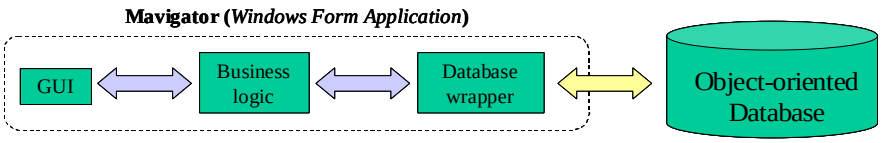

Figure 5. Architecture of the Mavigator prototype

The Mavigator prototype will utilize active extensions written in Microsoft C#. The functionality requires compiling and running source code (which will start particular extension) during execution (runtime) of the Mavigator. We have already recognized how to implement such a functionality in C#. The solution needs to define interface having methods to be invoked during user's activities and to determine a way to compile the user's code. Both problems are currently solved and we start to implement corresponding functionalities.

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops,
Agia Napa, Cyprus, October 25-29, 2004
Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822–833, 2004.

6 Evaluation

The SKGN prototype has been used for the Structural Fund Projects Portal thus we have some informal response from the users, generally very positive. SKGN, as a part of European project ICONS, has also been positively assessed by the Project Officers.

Mavigator will be tested according to prepared formal scenarios from the point of view of usability, efficiency, easy-of-use and expressive power. Two (or more) groups of users will get typical queries; sample queries can be found e.g. in [10, 23]. One of the groups will use the Mavigator, the rest other methods (such a textual query languages or other visual tools). The results will be judged from various points of view. More information about evaluating visual tools and graphical user interfaces can be found in [24, 25, 26].

7 Conclusion

This paper describes the Mavigator prototype and its visual metaphors that offer new quality for visual querying of object-oriented databases. Our ideas have been presented from the PhD thesis point of view. We have stressed the items, which should be included in the dissertation. We have also enumerated some open issues, which should be carefully investigated. We hope that the OTM PhD Symposium will be a great opportunity to discuss problems related to visual querying and our particular approach to solve them.

References

1. Object Data Management Group: The Object Database Standard ODMG, Release 3.0. R.G.G.Cattel, D.K.Barry, Ed., Morgan Kaufmann, 2000
2. Subieta K., Beeri C., Matthes F., Schmidt J.W.: A Stack-Based Approach to Query Languages. Proc. 2nd East-West Database Workshop, 1994, Springer Workshops in Computing, 1995, 159-180
3. Zloof M. M.: Query-by-Example: A Database Language. IBM Syst. Journal, 16(4), 1977, 324-343
4. Chimera, R.: Value Bars: An Information Visualization and Navigation Tool for Multi-Attribute Listings. Proc. ACM CHI ‘92, pp. 293-294, (1992).
5. Kumar, H., Plaisant, C., Shneiderman, B.: Browsing Hierarchical Data with Multi-Level Dynamic Queries and Pruning. IJHCI, vol. 46, pp. 103-124, (1997).
6. Catarci T.: What Happened When Database Researchers Met Usability. Information Systems 25(3), 2000, 177-212.
7. Trzaska M., Subieta K.: Structural Knowledge Graph Navigator For The Icons Prototype. Proc. of the IASTED International Conference on Databases and Applications (DBA 2004)

8. Trzaska, M., K.Subieta, K.: The User as Navigator. Proc. 8th East-European Conference on Advances in Databases and Information Systems (ADBIS), September 2004, Budapest, Hungary, to appear
9. Carey M.J., Haas L.M., Maganty V., Williams J.H.: PESTO: An Integrated Query/Browser for Object Databases. Proc. VLDB (1996) 203-214
10. Murray N., Goble C., Paton N.: Kaleidoscape: A 3D Environment for Querying ODMG Compliant Databases. In Proceedings of Visual Databases 4, L'Aquila, Italy, May 27-29, 1998
11. Murray N., Paton N.W., Goble C.A., Bryce J.: Kaleidoquery - A Flow-based Visual Language and its Evaluation. Journal of Visual Languages and Computing 11(2), 2000, 151-189
12. Fegaras L.: VOODOO: A Visual Object-Oriented Database Language For ODMG OQL. ECOOP Workshop on Object-Oriented Databases 1999, 61-72
13. Jones S., McInnes S.: A graphical user interface for boolean query specification. International Journal on Digital Libraries Special Issue on User Interfaces for Digital Libraries, 2(2/3):207–223, 1999
14. Shneiderman, B.: Visual user interfaces for information exploration. In Proceedings of the 54th Annual Meeting of the American Society for Information Science, pages379–384, Medford.NJ, 1991.Learned Information Inc.
15. Stolte Ch., Tang D., Hanrahan P.: Polaris: A System for Query, Analysis and Visualization of Multidimensional Relational Databases. IEEE Transactions on Visualization and Computer Graphics, Vol 8, No 1, January-March 2002
16. Cassel K., Risch T.: An Object-Oriented Multi-Mediator Browser. 2nd International Workshop on User Interfaces to Data Intensive Systems, Zürich, Switzerland, May 31 - June 1, 2001
17. Smith M., King P.: The Exploratory Construction Of Database Views. Research Report: BBKCS-02-02, School of Computer Science and Information Systems, Birkbeck College, University of London, 2002
18. Batini C., Catarci T., Costabile M.F., Levialdi S.: Visual Strategies for Querying Databases. Proc. of the IEEE Int. Workshop on Visual Languages, Japan, October 1991.
19. Derthick M., Kolojejchick J., Roth S.F.: An Interactive Visual Query Environment for Exploring Data, Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '97), ACM Press, (1997) 189-198
20. Bertin J.: Semiology of Graphics. Madison, Wis.: Univ. of Wisconsin Press, 1983. Translated by W. J. Berg
21. Nowell L.T.: Graphical Encoding for Information Visualization: Using Icon Color, Shape, and Size To Convey Nominal and Quantitative Data. PhD dissertation, Virginia Polytechnic Institute and State University, 1998
22. Matisse – a post-relational database. www.matisse.com
23. Chavda M., Wood P. T.: Towards an ODMG-Compliant Visual Object Query Language. In Proc. 23rd Int. Conf. on Very Large Data Bases, 1997, pp. 456-465
24. Ivory M. Y., Hearst M. A.: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. 33(4): 470-516 (2001)
25. Plaisant C.: The Challenge of Information Visualization Evaluation. In Proc. of Conf. on Advanced Visual Interfaces AVI'04 (2004) (to appear)
26. Rivadeneira W., Bederson B. B.: A Study of Search Result Clustering Interfaces: Comparing Textual and Zoomable User Interfaces. Human-Computer Interaction Lab / Univ. of Maryland, Report no HCIL-2003-36, October 2003