Visual Information Expert Workshop, Paris, France, April 24 – 25.2006
Springer LNCS 4370, ISBN 978-3-540-71026-4

Evaluation of the Mavigator

Mariusz Trzaska[*], Kazimierz Subieta[#*]

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

Abstract. Mavigator is a graphical querying and browsing tool dedicated to naive users (computer non-professionals). It allows them to retrieve and analyse information from various data sources, in particular, from object-oriented and XML-oriented databases, providing a corresponding wrapper is implemented. Mavigator key concepts related to information retrieval include: intensional navigation, extensional navigation, and persistent baskets for recording temporary and final results. A basic problem related to end-user visual interfaces concerns final presentation and further processing of retrieval results. The Mavigator Active Extensions (AE) module allows the programmer to extend ad hoc the existing core functionalities through a program written in C#. Thus the retrieved data can be presented and analysed in any conceivable visual form. Another novel feature of Mavigator is Virtual Schemas, which make it possible to customize database schema, in particular, changing some names, adding virtual associations or hiding some classes. Virtual Schemas allow creating a customized version of an existing database schema and navigate within the database according to this schema. In this paper we present results of usability tests, which have been conducted within three independent student groups. The results of the tests suggest improvements to the prototype.

1. Introduction

According to [1] and [2], the application's evaluation process has four main aims:

- Discovering major problems that could result in a human error or lead to frustration of the user,
- Reducing training time,
- Increasing performance and efficiency,
- Improving user's satisfaction from using the software.

The usability of an application can be defined as an ability to satisfy user's needs related to the application. As noted in [3] there are two fundamental approaches to usability:

- By principles. This means choosing such usability principles, which will be adequate for a particular kind of an application and user,
- By evaluation. This approach requires evaluation by the users, which means that the whole application (or at least some part of them) must be developed. The easy part of the method is criticizing current solutions. The hard part is deciding what to change to improve it.

We believe that developers of applications have to combine two above approaches. During the entire application's creation process, all known usability principles must be taken into consideration. Still, after developing a beta version, the evaluation with users must be eventually conducted. In this paper we present the research results along this line.

In [4] Plaisant distinguishes four thematic areas of evaluation:

- Controlled experiments comparing design elements. Such studies should compare specific widgets and mappings of information to a graphical display,
- Usability evaluation of a tool. Such studies should provide feedback on the problems that the users encounter with a tool and should explain how the designers have to refine the design,
- Controlled experiments comparing two or more tools. These studies usually try to compare a novel technique with the state of the art.
- Case studies of tools in realistic settings. This is the least common type of studies. The advantage of case studies is that they report on users in their natural environment doing real tasks, demonstrating feasibility and in-context usefulness. The disadvantage is that they are time consuming to conduct, and the results may not be replicable.

Our first prototype SKGN has been used in the European project ICONS. It was informally evaluated during using for the Structural Fund Projects Portal implemented at the top of ICONS. Hence we have some informal response from the users, generally very positive. However, we need more formal evaluation, thus we have decided to conduct usability evaluation of the tool (the second area of evaluation). The procedure is generally enough to utilize them in evaluating almost every kind of graphical user interfaces, including systems employing pixellisation paradigm.

To fully understand the evaluation process some general information regarding the Mavigator must be presented. Hence, after presenting related work (Section 2), next sections briefly discuss key concepts of the Mavigator: information retrieval capabilities (Section 3), Active Extensions (Section 4) and Virtual Schemas (Section 5). Section 6 is dedicated to the evaluation itself. Section 7 concludes.

2. Related Work

Related solutions can be analysed from three points of views: methods of modifying application's functionalities, the way of information retrieval and utilization of database views. However due to the limited space we will focus on the second topic only (more information can be found in [5, 6]).

Roughly, visual metaphors to information retrieval can be subdivided into two groups: based on graphical query languages and graphical browsing interfaces. The subdivision is not fully precise because many systems have features from both groups. An example is Pesto [7] having possibilities to browse through objects from a database. Otherwise to Mavigator, the browsing is performed from one object to a next one. 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 less professional users and some kinds of retrieval tasks.

Typical visual querying systems are Kaleidoscape [8] and VOODOO [9]. Both are declared to be visual counterparts of ODMG OQL 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. We find it very useful and intuitive thus we have adopted it to our metaphor.

A typical example of a browsing system is GOOVI [10]. 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 Watson [11], 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. 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. Information retrieval capabilities

Mavigator is made up of three metaphors utilized for information retrieval: intensional navigation, extensional navigation and persistent baskets. The user can combine these metaphors in an arbitrary way to accomplish a specific task.

Intensional 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 intensional 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 short description of the methods (more information can be found in [12]).

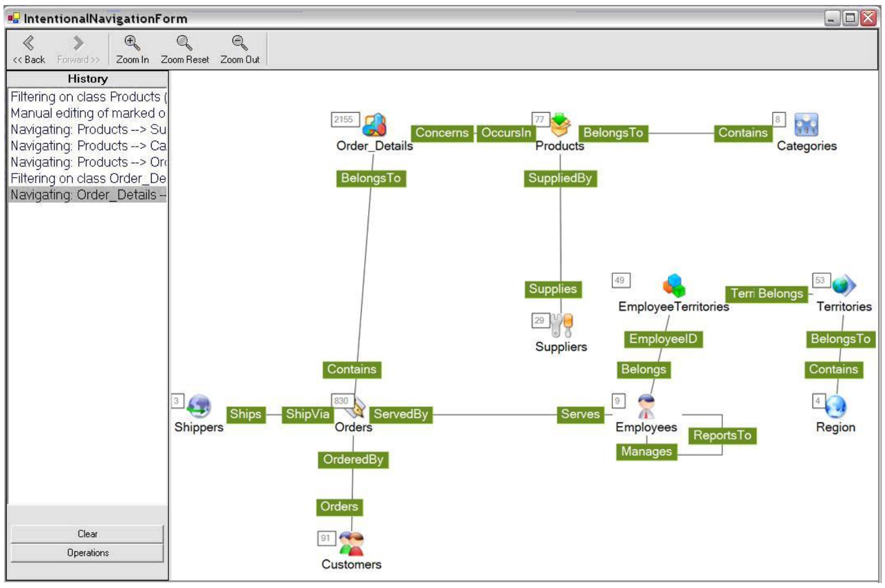
3.1. Intensional navigation

Intensional navigation utilizes a database schema graph. Figure 1 shows a window containing a database scheme graph of the Northwind sample. The 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 and the number of all objects in the class,
- Edges, which represent semantic associations among objects (in UML terms),
- Labels with names of association roles. They are understood as pointers from objects to objects (like in the ODMG standard, C++ binding).

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.
- Manual selection. Using special labels 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.



Filtering on class Products (
Manual editing of marked o
Navigating: Products --> Su
Navigating: Products --> Ca
Navigating: Products --> Or
Filtering on class Order_De
Navigating: Order_Details --

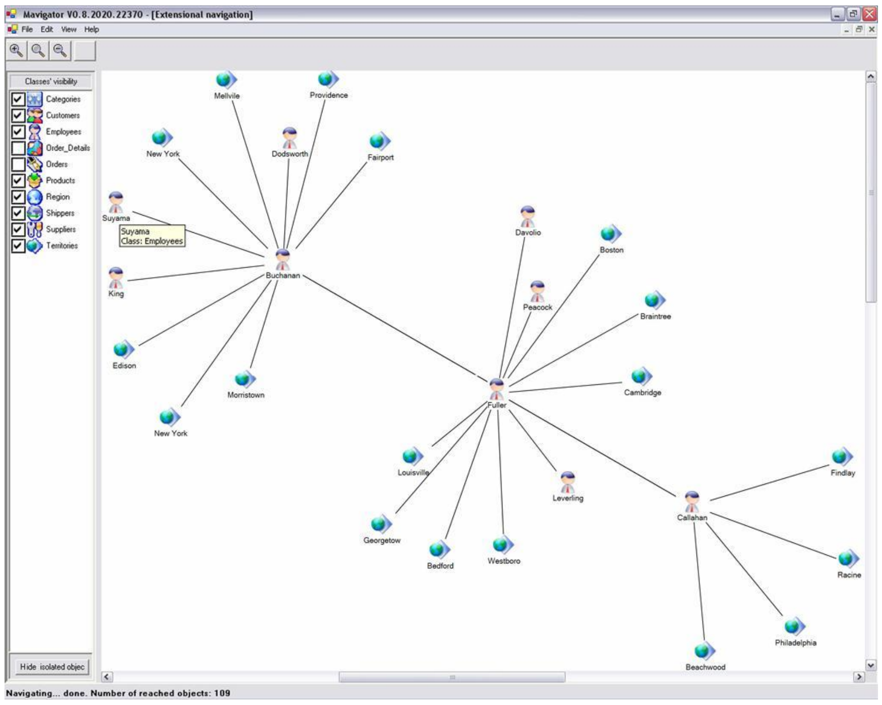
Figure 1. Intensional navigation window

- Navigation from marked objects of one class, through a selected association role, to objects of another class. An object from a target class becomes marked if there is an association link to the object from a marked object in the source class.

- Basket activities (see further).
- Active extensions (see further).

Intensional 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.

3.2. Extensional navigation



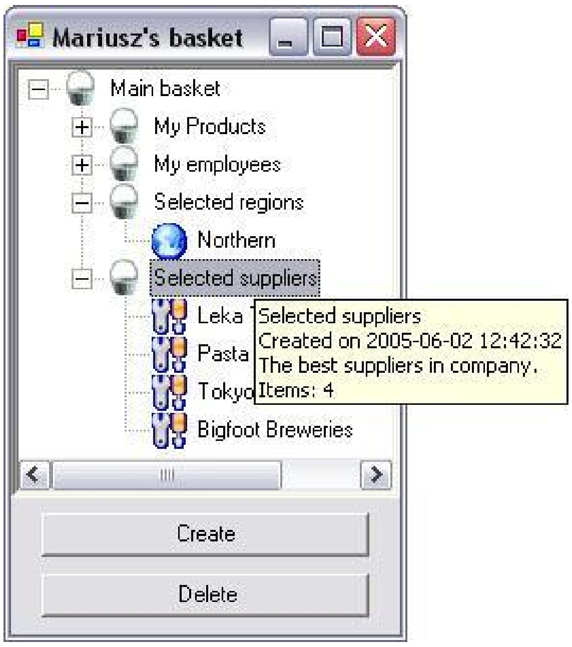
Navigating... done. Number of reached objects: 109

Figure 2. Extensional navigation window.

Extensional navigation takes place inside extensions of classes (Figure 2). 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



Selected suppliers
Created on 2005-06-02 12:42:32
The best suppliers in company.
Items: 4

Figure 3. Basket visualization.

Baskets are persistent storages of search results. They store two kinds of entities: unique object identifiers (OIDs) (seen as object labels) and sub-baskets. The hierarchy of baskets is especially useful for information categorization and keeping order. 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 is assigned to a particular user. At the end of a user session all the baskets are stored in the database.

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

4. Active Extensions

Mavigator already employees some information retrieval metaphors (see section 3), which are powerful and yet easy-to-use, so we have decided to provide a way to add new functionalities operating only on a query result. The approach does not complicate the entire application, but guarantees sufficient flexibility. Thus our solution requires collaboration of an end user with a programmer who will write the code accomplishing the required functionalities.

The current Mavigator prototype uses Microsoft C# as a language for Active Extensions. A programmer is aware of the Mavigator metadata, which allows him/her to write a source code of the required functionality in C#. Writing the Active Extension source code is done in the Mavigator's special editor. Once a programmer compiles the code, a particular Active Extension is ready to use (without stopping Mavigator). Then the end user is supported with one click button causing execution of the written code. It processes the query result (see Section 3.1) or objects recorded in a user basket (see Section 3.3). The functionality of such programs is unlimited and can be a simple one (i.e. calculating average value of selected attributes), through objects exporters (i.e. to XML format), up to more complicated ones like Active Projections.

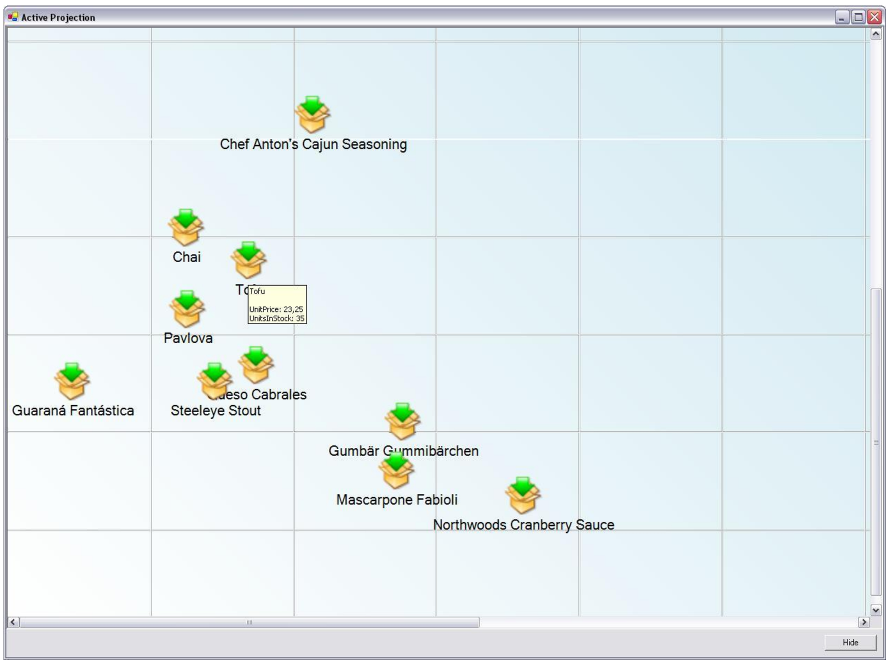


Figure 4. Active projections.

Active Projections (Figure 4) allow visualizing a set of objects where the position (x and y coordinates) of each of them is based on value of particular objects' attributes. Current implementation uses two axes (2D), which allow visualizing dependencies of two attributes. Besides the visual analysis of objects dependencies it is also possible to utilize projections in more active fashion. Object taken from a basket can be dropped on a projection's surface, which causes right (based on attributes values) placement. It is also possible to perform a reverse action: drag an object from the surface onto a basket, which causes recording the object in the basket. More information can be found in [12].

5. Virtual Schemas

A virtual schema (a view) is a mapping of a database schema according to needs of the current user. A virtual schema exists as a definition only, no physical mapping of data is performed. According to the fundamental transparency requirement, the user uses a virtual schema in the same manner as an original database schema. Virtual schemata have low resource demands, thus Mavigator is capable to support user's work with many virtual schemas in the same retrieval session.

Generally, Virtual Schemas allow creating customized database views consisting of the following elements:

- Virtual associations, which reflect any dependencies among classes,
- Virtual attributes, which describe objects' properties.
- Classes, which are counterparts of physical collections from the database.

More detailed discussion of the topic can be found in [6].

6. Evaluation

Mavigator's evaluation has been conducted by the 6 subjects in three groups (because of problems with scheduling). All of them were students from the Polish-Japanese Institute of Information Technology (different departments). All surveys were in Polish and based on [13, 14].

6.1. Procedure

At the beginning, subjects have to answer questions about their experience and knowledge. All answers were from range 0 (lack of knowledge) to 10 (expert). Below we have enumerated all of them (question 1.1 is just an id of the subject):

1.2 Object-oriented database concepts (classes, associations, etc.),
1.3 Microsoft Access experience,
1.4 Textual query language experiences (i.e. SQL, OQL),
1.5 Visual information retrieval system experience,
1.6 Programming language experience.

Then, all subjects were trained on the Mavigator prototype. The training program consisted of short introduction with description of motivations behind Mavigator, discussion of Mavigator's key concepts and detailed instructions on using the prototype. This includes demonstration of using particular techniques like filtering, navigating, etc.

After the training, subjects have to find answers for following query questions (working with the "Northwind" database mentioned earlier):

1. Find all employees with the first name "Robert".
2. Find all employees with the first name "Robert" or "Nancy".
3. Find the name of an employee who manages the territory "Cambridge".

4. Find names and prices for all products from supplier "Leka Trading" where prices are more than $18.00.
5. Find employees who served orders sent to Mexico and manage territories located in the "Northern" region.
6. Find average price of the products sent to the Mexico.
7. Find the cheapest product among those ones, which are in stock.
8. Analyze the "Seafood" products, to find out which of them are not so much on the stock.

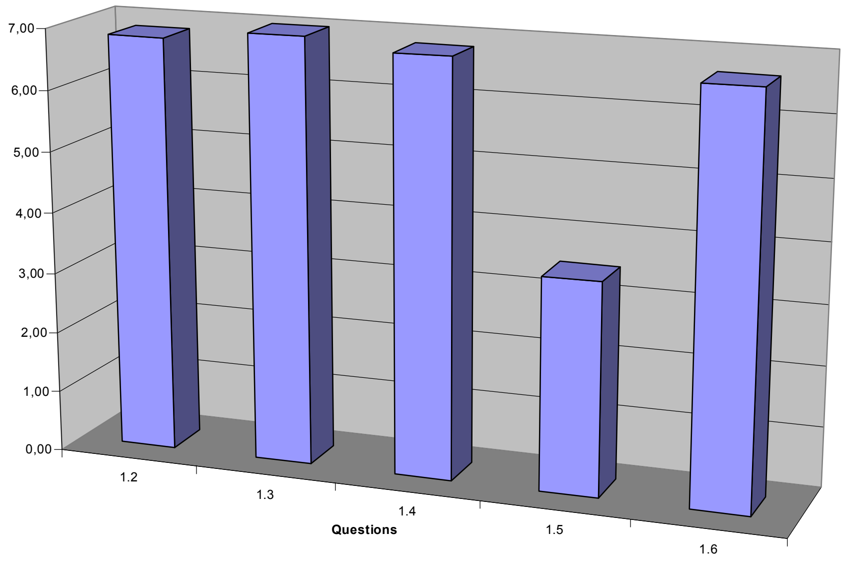


Figure 5. Average information about subjects.

For each query question, each subject has to fulfil survey with the following items (2.1 is an id of the subject, 2.2 is the number of the query question):

2.3 Difficulty of the query question – from 1 (easy) to 10 (very hard),
2.4 Success - in percents,
2.5 Time to completion – in minutes,
2.6 Remarks.

At the end of evaluating, after solving all query problems, each user answers summary questions about the metaphors, prototype, etc:

3.1 Comprehensibility – from 1 (confusing) to 10 (clear),
3.2 Ease of use – from 1 (difficult) to 10 (easy),
3.3 Speed of use – from 1 (slow) to 10 (fast),
3.4 Performance – from 1 (slow) to 10 (fast),
3.5 Overall satisfaction – from 1 (terrible) to 10 (wonderful),
3.6 Remarks,
3.7 Ideas for improvement,

3.8 Ideas for new Active Extensions.

6.2. Results

Figure 5 shows average values given by subjects describing themselves. Bars refer to the questions enumerated in section 6.1. According to the answers, all subjects have been quite familiar with database concepts (question 1.2), MS Access (question 1.3), textual query languages (question 1.4) and programming language (question 1.6). However, most of them were not familiar with graphical user interfaces (question 1.5). From the testing point of view, maybe it would be better to find subjects less familiar with these topics. However, we have to take into account that some of the answers could be a little bit exaggerated. This might be caused by the fact that subjects were students (and the administrator of the experiment was their teacher), who should be familiar with the mentioned terms.

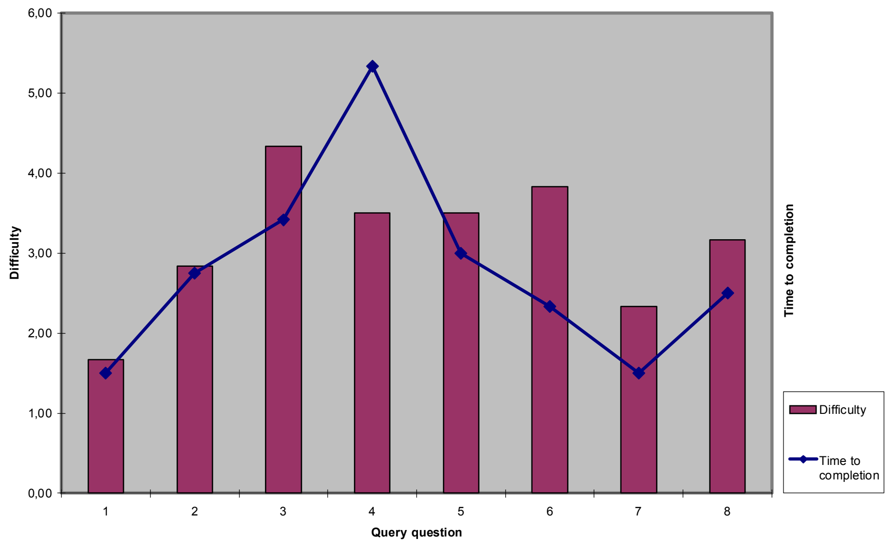


Figure 6. Chart showing dependence between questions' difficulty and time for completion.

Figure 6 presents chart showing two kinds of information (for each query question):

- Average difficulty of the query questions (bars),
- Average time to completion (line).

As it can be seen, queries, which has been judged as harder, take more time to complete (except the fourth one). Also, almost all queries (without one – for one subject), have been completed in 100 percent (not shown on the chart). The shortest time to complete was 1.5 min. (for number 1 and 7), the longest one was 5.5 min. (for number 4) and an average time to complete was about 2 minutes and 45 seconds.

According to the subjects' answers, the harder question was number 3 and then number 6. However, even the hardest ones, still have been judged as easier than medium (less then 5 in the 10 degrees scale).

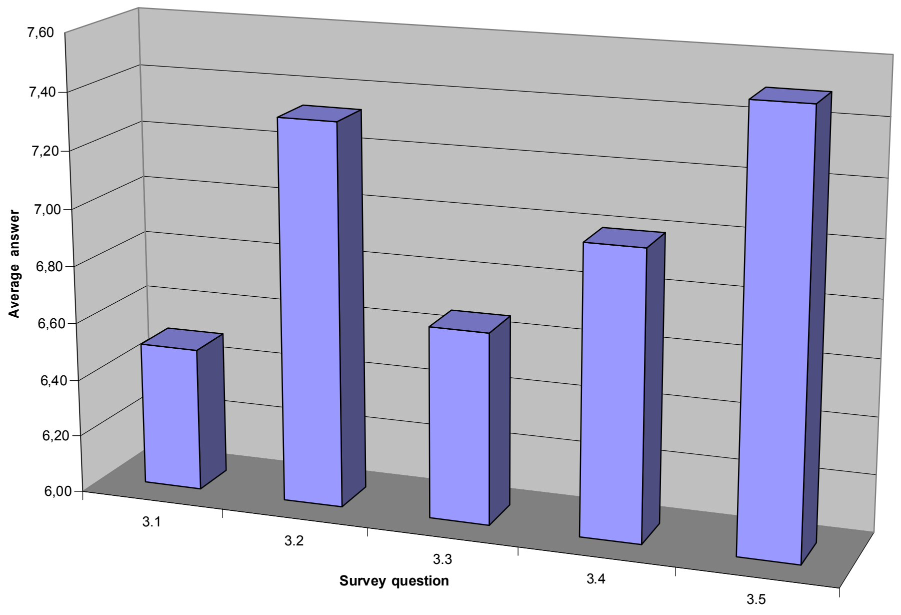


Figure 7. Chart showing average answers about Mavigator.

Figure 7 shows average answers (to the questions from section 6.1), given by the subjects, about Mavigator's prototype. Roughly speaking, all answers are positive (more then 5 in the tenth degrees scale).

Comprehensibility (question 3.1), which is a very important factor in graphical user interfaces, has average value 6.5 with minimum at 4 (one subject) and maximum at 8 (also one subject). Most subjects give 7/10.

Another important factor, which has been judged by the subjects, is ease of use (question 3.2). In this case, marks vary from 6, through 7 and 8 up to the 9. Average value was 7.3/10, which is quite high.

Speed of use (question 3.3), which is different from speed of working (performance – 3.4), describes things like the way of using options, running actions, etc. This factor is related in some way to the comprehensibility, and has achieved average value about 6.6/10.

Speed of working (question 3.4) is not so important in case of prototypes. Moreover performance of the information retrieval tools (including Mavigator) is tightly connected with the performance of the data source. The current prototype works with the ODRA experimental database server, which has not been optimized yet. However, in this field Mavigator has also gotten quite a good result: 7/10.

The last factor, which has been assessed, overall satisfaction of the user. This is very subjective judgment and has big impact on the user's decision: do I really want to use the tool? Fortunately, Mavigator's users give them average mark about 7.5/10 with minimum at 7 and maximum at 9. Such a result bears good testimony to the ideas and metaphors behind Mavigator:

6.3. Users' Remarks

Subjects have the right (and they have been encouraged to) to make comments and remarks. All of them have been carefully analyzed. Most of them were related to the graphical user interface (and have been taken into account) and not to the metaphors itself:

- Items' Lists. One subject noticed that a list of attributes during defining a single predicate is not sorted any way. As a result all lists (of items of any kind) appearing in the prototype have been analyzed and sorted alphabetically.
- Showing Objects. Another subject gave attention that one of the most frequently performed operations is showing marked objects. Thus, it would be nice to have an opportunity to perform it very quick. Hence, a new way of showing marked objects has been added: when a user double clicks on the class icon, a list of marked object is shown.
- Showing Basket. Next problem was connected with showing basket's window. When the window has been shown, and then another window covered them, choosing Show basket from the menu, has no effect. It has been fixed by bringing the basket's window to the front (after selecting appropriate option form the menu).
- Working with a Single Predicate. When a user defines a single predicate (for filtering some objects), there is a necessity to enter some values. One of the subjects suggests that application should prompt values of the selected attributes read from existing objects. The idea is obviously good. However, its implementation could lead to serious performance problems. This is caused by the fact that prompting values requires reading all objects from the database (or at least all values of the particular attribute). This is technically possible, but the performance overhead makes it questionable.
- Marking objects with filtering. During filtering objects, one of the subjects reported that filtering system does not work. After short investigation it comes out that the user is filtering from 0 marked objects, which leads to marking 0 objects. As a result, a dedicated message has been added, which informs the user about the situation.

6.4. Procedure Remarks

As mentioned previously, Mavigator's evaluation procedure is quite general and could be utilized in many different kinds of graphical information retrieval/analysis tools. In particular, the procedure, without a big effort, could be tailored for evaluating systems which employs pixellisation paradigm. Following items contain a brief discussion of such utilization:

- The first point of the procedure (questions from 1.2 – 1.6) collects information about the subject. Thus, such information are useful during evaluation all kinds of systems. Based on the answers we can judge the results of the entire evaluation.
- Next task included training of the prototype which is necessary in case of all new systems. After that subjects have to find answers to some problems. In case of Mavigator it was some kind of data related queries. In case of other analysis tools it would be other data related activities i.e. finding some relations among information, etc.
- Questions from point 2 give information about difficulty of the problem, time to completion, etc (they should be answered for each query problem). All of them could be applied with minor changes.
- And the last group of questions (3.1 – 3.8) stores some overall opinion about the system. Such questions are necessary during evaluating all kind of computer systems.

7. Conclusions and future work

We have presented the Mavigator and its evaluation. Conducted surveys satisfied two important evaluation's targets:

- Allowed improving the prototype,
- Proved rightness of the Mavigator's ideas and metaphors.

Overall high marks, issued by the prototype's users confirm Mavigator's high usability and easy-in-use.

The prototype offers new quality in two main areas. The first one is extending existing application's functionalities. Active Extensions, which use fully-fledged programming language, make it possible to create any kind of additions to Mavigator's core functions. The second area contains Virtual Schemas, which allow creating customized version of existing database schema.

As a continuation of our research, we have started a work related with connecting the Mavigator with the eGov-Bus virtual repository. This data source is developed as a part of the EC project called eGov-Bus[1].

We also plan investigation on adding new functionalities, which will make our system more powerful and easy-to-use.

[1] Advanced Government Information Service Bus (eGov-Bus, IST 26727 STP) is a project supported by the EC as a part of the Sixth Framework Programme.

References

1. Shneiderman B., Plaisant C.: Designing the user interface: Strategies for effective human-computer interaction (4th Ed). Reading, MA: Addison-Wesley, 2003.
2. Norman K. L., Panizzi E.: Levels of Automation and User Participation in Usability Testing, University of Maryland, Laboratory for Automation Psychology and Decision Processes, Technical Report: LAP-2004-01, HCIL-2004-17, 2004.
3. Murphu N.: Principles of User Interface Design. Internet Appliance Design Article, December 2000, http://www.embedded.com/2000/0012/0012ia1.htm.
4. Plaisant C.: The Challenge of Information Visualization Evaluation. In Proc. of Conf. on Advanced Visual Interfaces AVI'04 (2004).
5. Trzaska M., Subieta K.: Active Extensions in a Visual Interface to Databases, Fourteenth International Conference on Information Systems Development (ISD´2005), Kluwer/Plenum Press, 14-17 August, 2005, Karlstad, Sweden.
6. Trzaska M.: Virtual Schemas in Visual Interfaces to Databases, I Krajowa Konferencja Naukowa "Technologie Przetwarzania Danych", pp. 361 - 371, 26-28 September, 2005, Poznan, Poland.
7. Carey M.J., Haas L.M., Maganty V., Williams J.H.: PESTO: An Integrated Query/Browser for Object Databases. Proc. VLDB (1996) 203-214
8. Murray N., Goble C., Paton N.: Kaleidoscape: A 3D Environment for Querying ODMG Compliant Databases. In Pro. of Visual Databases 4, L'Aquila, Italy, May 27-29, 1998
9. Fegaras L.: VOODOO: A Visual Object-Oriented Database Language For ODMG OQL. ECOOP Workshop on Object-Oriented Databases 1999, 61-72
10. 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
11. 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
12. Trzaska M., Subieta K.: Usability of Visual Information Retrieval Metaphors for Object-Oriented Databases. Proceedings of the On The Move Federated Conferences and Workshops (DOA, ODBASE, CoopIS, PhD Symposium), Springer Lecture Notes in Computer Science (LNCS 3292), pp. 822-833, October 25-29, 2004, Larnaca, Cyprus.
13. Holyer A.: Methods for Evaluating User Interfaces. Cognitive Research Paper No. 301, School of Cognitive and Computing Sciences, University of Sussex, Brighton, 1993.
14. North C. L., A User Interface for Coordinating Visualizations Based on Relational Schemata: Snap-Together Visualization. PhD Dissertation, Graduate School of the University of Maryland, College Park, 2000.